

MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII

# ИНФОРМАТИКА

**Учебник для 8-го класса**

АНАТОЛ ГРЕМАЛСКИ • ЮРИЕ МОКАНУ • ИОН СПИНЕЙ  
ЛУДМИЛА ГРЕМАЛСКИ



**Acest manual este proprietatea Ministerului Educației, Culturii și Cercetării.**

Manualul școlar a fost realizat în conformitate cu prevederile Curriculumului la disciplină, aprobat prin Ordinul Ministerului Educației, Culturii și Cercetării nr. 906 din 17 iulie 2019. Manualul a fost aprobat prin Ordinul Ministerului Educației, Culturii și Cercetării nr. 1048 din 28.09.2020, urmare a evaluării calității metodică-științifice.

Editat din sursele financiare bugetare.

**Comisia de evaluare:**

**Coordonator:** *Natalia Schițco*, profesor, grad didactic superior, LT „Socrate”, Chișinău;  
*Svetlana Brînză*, profesor, grad didactic superior, LT „N.M. Spătaru”, Chișinău;  
*Viorica Juc*, profesor, grad didactic superior, LT „Ion Creangă”, Chișinău;  
*Olga Turchin*, profesor, grad didactic unu, LT „V. Suhomlinski”, Edineț;  
*Diana Bagrin*, profesor, grad didactic unu, LT „Da Vinci”, Chișinău

Denumirea instituției de învățământ _____				
Manualul a fost folosit: _____				
Anul de folosire	Numele, prenumele elevului	Anul de studii	Aspectul manualului	
			la primire	la returnare

Dirigintele verifică dacă numele, prenumele elevului sunt scrise corect.

Elevii nu vor face niciun fel de însemnări în manual.

Aspectul manualului (la primire și la returnare) se va aprecia cu unul dintre următorii termeni: *nou, bun, satisfăcător, nesatisfăcător*.

**Responsabil de ediție:** Larisa Dohotaru  
**Redactor:** Tatiana Șarșov  
**Corectori:** Larisa Nosacenco, Alefina Olari

**Redactor tehnic:** Nina Duduciuc  
**Machetare computerizată:** Vitalie Ichim  
**Copertă:** Romeo Șveț

ÎNȚEPRINDEREA  
EDITORIAL-POLIGRAFICĂ

**ȘTIINȚA**

str. Academiei, nr. 3; MD-2028, Chișinău, Republica Moldova  
tel.: (+373 22) 73-96-16; fax: (+373 22) 73-96-27  
e-mail: prini\_stiinta@yahoo.com;  
www.editurastiinta.md

Toate drepturile asupra acestei ediții aparțin Întreprinderii Editorial-Poligrafice Știința.

**Descrierea CIP a Camerei Naționale a Cărții**

**Informatica:** Учебник для 8-го класса / Анато́л Гремалски, Ю́рие Мокану, Ио́н Спи́ней, Лудми́ла Гремалски; comisia de evaluare: Natalia Schițco (coordonator) [et al.]; Ministerul Educației, Culturii și Cercet. –

Ch.: Știința, 2020 (Tipogr. "Balacron"). – 152 p.: fig., tab.

Proprietate a Min. Educației, Culturii și Cercet.

ISBN 978-9975-85-243-2

Imprimare la Tipografia **BALACRON SRL**

str. Calea Ieșilor, 10, MD-2069, Chișinău, Republica Moldova, comanda nr. 833

© *Anatol Gremalschi, Iurie Mocanu, Ion Spinei, Ludmila Gremalschi*. 2012, 2016, 2020

© Traducere din limba română: *Arcadie Malearovici, Irina Ciobanu*. 2012, 2016, 2020

© Întreprinderea Editorial-Poligrafică Știința. 2012, 2016, 2020

## СОДЕРЖАНИЕ

<b>Глава 1. Редактирование текстов</b>	<b>5</b>
1.1. Приложение <i>Word</i>	5
1.2. Форматирование символов	9
1.3. Форматирование абзацев	14
1.4. Форматирование страниц	19
1.5. Списки и таблицы	23
1.6. Вставка объектов	27
1.7. Форматирование изображений	33
1.8. Объектно-ориентированная графика	38
1.9. Диаграммы	44
1.10. Проверка правописания	48
1.11. Вставка формул	54
1.12. Стили и шаблоны	56
1.13. Комбинированная корреспонденция	63
<b>Глава 2. Алгоритмы</b>	<b>67</b>
2.1. Алгоритмы и исполнители	67
2.2. Субалгоритмы	73
2.3. Циклические алгоритмы. Цикл со счетчиком	78
2.4. Циклические алгоритмы. Цикл с условием	83
2.5. Алгоритмы с разветвлениями	88
2.6. Алгоритм работы компьютера	91
2.7. Основные сведения об алгоритмах	93
<b>Глава 3. Реализация алгоритмов в средах текстуального программирования</b>	<b>97</b>
3.1. Концепция действия	97
3.2. Выражения	98
3.3. Вычисление выражений	100
3.4. Тип выражений	101
3.5. Оператор присваивания	105
3.6. Оператор <i>вызова процедуры</i>	107
3.7. Вывод алфавитно-цифровой информации на экран	108
3.8. Ввод данных с клавиатуры	111
3.9. Пустой оператор	114
3.10. Оператор <i>if</i>	114
3.11. Оператор <i>case</i>	117
3.12. Оператор <i>for</i>	120
3.13. Составной оператор	123
3.14. Оператор <i>while</i>	125
3.15. Оператор <i>repeat</i>	129
<b>Глава 4. Редактирование изображений</b>	<b>132</b>
4.1. Цифровые изображения	132
4.2. Цветовые модели	134
4.3. Форматы графических файлов	136
4.4. Практические занятия: оборудование и форматы графических файлов	139
4.5. Приложение <i>Paint 3D</i>	140
4.6. Практические занятия: Обработка трехмерных изображений	144
4.7. Слои и каналы цветов	145
4.8. Приложение <i>GIMP</i>	148
4.9. Практические занятия: Обработка растровых изображений	151
4.10. Рекомендуемые проекты	152

## Дорогие друзья,

Информатика меняет нашу жизнь иногда совершенно неожиданным образом даже для знатоков в этой области. Чтобы владеть ситуацией, быть конкурентоспособными и использовать плоды этой науки, мы должны знать и уметь применять весь арсенал информационных методов и приемов.

Производительность современных компьютеров — скорость работы, объем внутренней памяти, состав и характеристики периферийных устройств — с каждым годом практически удваиваются. Такой же впечатляющий прогресс отмечен в области коммуникаций, что позволяет объединить все компьютеры в глобальную сеть, содержащую огромное количество информации. Эта информация производится людьми и для людей. Чтобы эффективно использовать накопленную информацию и способствовать созданию новых знаний, каждый человек должен формировать и развивать свою информационную культуру и алгоритмическое мышление. Это требует досконального знания основных концепций информатики и способности разрабатывать алгоритмы для решения задач, с которыми человек сталкивается в своей повседневной жизни и профессиональной деятельности.

Если в начале прошлого века, чтобы соответствовать вызову времени, каждому члену общества требовалось умение читать, то сегодня, в начале нового тысячелетия, каждому человеку всё настойчивей требуется компьютерная грамотность. Не зря термин «сертификат компьютерной грамотности» появился во многих языках мира. По последним данным, без такого сертификата большинство современных профессий, в том числе традиционных, становятся недоступными.

Цель данного учебника — дать ученикам возможность приобрести знания, необходимые для автоматической обработки документов с использованием текстовых редакторов и для формирования алгоритмического мышления.

Приложения для обработки текстов, являющиеся наиболее распространенными компьютерными программами, используются для ввода данных, вставки мультимедийных объектов и представления информации в привлекательной форме. Как и в случае изучения других компьютерных программ, текстовые редакторы являются мощным инструментом, который помогает нам узнать методы хранения и обработки информации, организации данных в таблицах, представления их в виде диаграмм. Теоретические и практические знания в области приложений для обработки текстов будут реально полезны не только после окончания гимназии, но и в процессе изучения других учебных дисциплин, таких как математика, физика, химия, география, история и др.

Формирование алгоритмического мышления предполагает понимание таких понятий, как: исполнитель и алгоритм, знание форм представления и свойств алгоритмов. Теоретический и практический материал, включенный в учебник, поможет вам сделать первые шаги в мире алгоритмов, разработать собственные компьютерные программы.

Для развития призвания каждого ученика в учебник включены модули по выбору: реализация алгоритмов в средах текстового программирования и обработка изображений. Изучение этих модулей будет осуществляться в основном посредством самостоятельного обучения с помощью онлайн-уроков, посредством различных практических занятий, путем разработки оригинальных проектов, тесно связанных со школьной жизнью, с местом, в котором вы живете.

*Желаем вам успехов!*

**Авторы**

## РЕДАКТИРОВАНИЕ ТЕКСТОВ

### 1.1. Приложение Word

Ключевые термины:

- документ
- структура документа
- окно документа
- панель инструментов

Приложение **Word** работает со специальными файлами, называемыми *документами*.

**Документ** представляет собой составной объект, состоящий из более простых объектов — текста, таблиц, изображений, звуковых и видеофрагментов, которые обрабатываются как единое целое.

Элементы документа образуют иерархическую структуру (рис. 1.1).

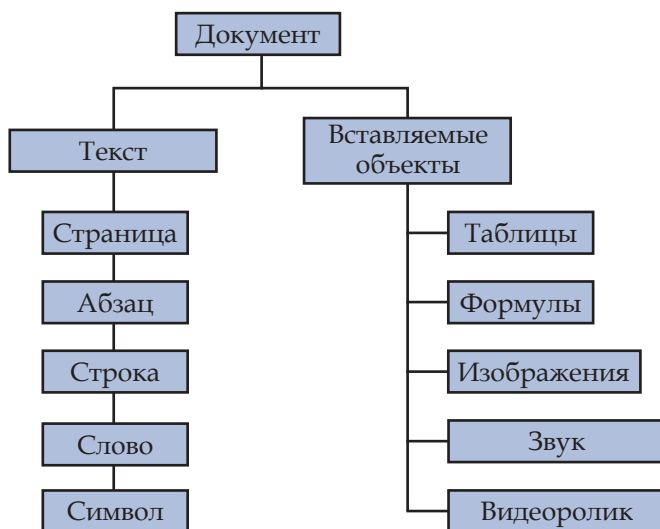


Рис. 1.1. Иерархическая структура документа

Каждый объект, входящий в состав документа, характеризуется определенными **свойствами**. Эти свойства можно изменять с помощью инструментов

приложения **Word**. Например, текст характеризуется типом и размером использованных символов, таблица — количеством строк и столбцов, рисунок — размерами и местоположением на странице. С помощью соответствующих инструментов пользователь может изменять вид и размеры выбранных символов, вставлять и удалять строки и столбцы в таблицах, увеличивать и уменьшать рисунки. Окно приложения **Word** представлено на рисунке 1.2.

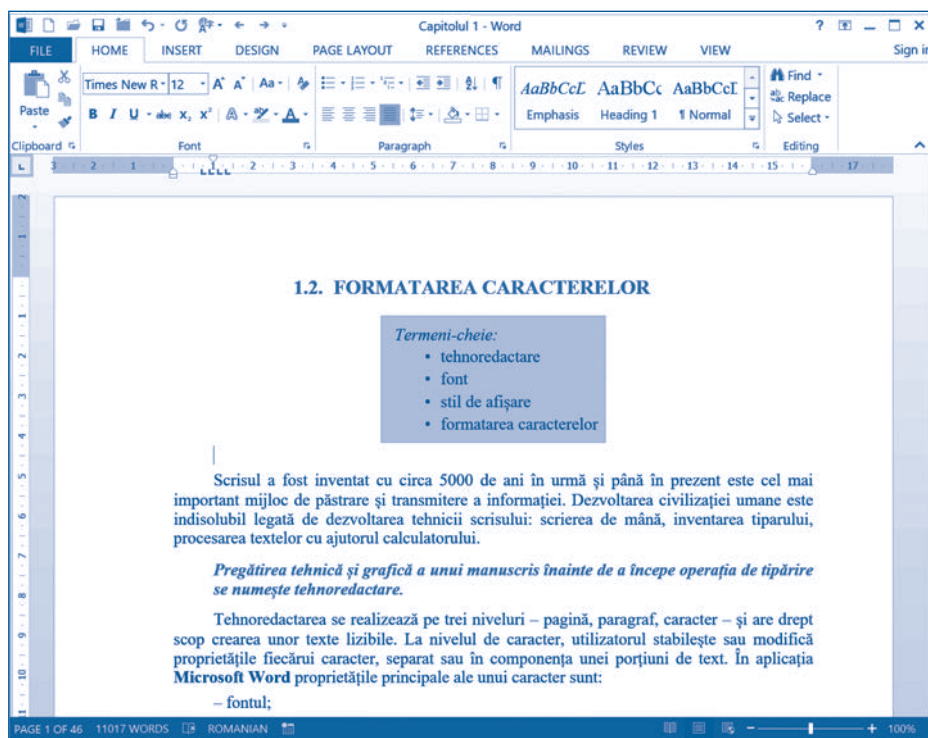


Рис. 1.2. Окно приложения **Word**

В окне приложения **Word** содержатся следующие элементы:

- строка заголовка;
- лента меню и закладки со стандартными инструментами;
- рабочая область;
- строка состояния.

Внутри рабочей области находятся:

- область текста;
- вертикальная панель прокрутки;
- горизонтальная панель прокрутки;
- вертикальная линейка;
- горизонтальная линейка.

В зависимости от версии, файлы, создаваемые с помощью приложения **Word**, имеют расширение **.doc** (документ) или **.docx**. Открывать и сохранять документы можно с помощью команд **New** (Новый), **Open** (Открыть), **Close** (Закрыть), **Save** (Сохранить), **Save As** (Сохранить как) в меню **File**. Быстрый

доступ к некоторым командам выполняется нажатием соответствующих кнопок на закладках с инструментами. Назначение каждой кнопки можно узнать, удерживая над ней указатель мыши.

Как и в приложении **Notepad**, текст любого документа может быть введен с клавиатуры. Текст редактируется с помощью команд **Cut** (Вырезать), **Copy** (Копировать), **Paste** (Вставить), **Clear** (Очистить), **Find** (Найти), **Replace** (Заменить) в меню **Home** (Главная) или при помощи соответствующих кнопок на закладке с инструментами.

В отличие от программы **Notepad**, приложение **Word** автоматически переносит на новую строку слова, не поместившиеся в текущей строке. Клавиша <Enter> нажимается только в том случае, если необходимо ввести символ «конец абзаца». Аналогичным образом при завершении текущей страницы автоматически выполняется переход на новую страницу.

## Вопросы и упражнения

- 1 Какие объекты могут содержать документы? Приведите примеры.
- 2 Чем характеризуются объекты, входящие в состав документов, создаваемых с помощью редактора текстов? Приведите примеры.
- 3 Укажите на *рис. 1.2* элементы окна **Word**.
- 4 **ИССЛЕДУЙТЕ!** Для чего предназначена лента меню? Какие инструменты содержат закладки каждого элемента меню?
- 5 Назовите команды, необходимые для открытия и сохранения документов. В каких случаях они используются?
- 6 Назовите команды для редактирования текста: стирания, копирования, перемещения, замены и т. п.
- 7 Каким образом обеспечивается быстрый доступ к командам меню? Приведите примеры.
- 8 **ИССЛЕДУЙТЕ!** Определите названия всех кнопок в закладках инструментов.
- 9 **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Создайте в приложении **Notepad** тексты **Otrocestvo**, **Parus** и **Alisa**. Сравните способы отображения текста в приложениях **Notepad** и **Word**.

### ОТРОЧЕСТВО

Рано, рано утром безжалостный и, как всегда бывают люди в новой должности, слишком усердный Василий сдергивает одеяло и уверяет, что пора ехать и всё уже готово. Как ни жмешься, ни хитришь, ни сердишься, чтобы хоть еще на четверть часа продлить сладкий утренний сон, по решительному лицу Василия видишь, что он неумолим и готов еще двадцать раз сдернуть одеяло, вскакиваешь и бежишь на двор умываться. В сенях уже кипит самовар, который, покрасневшись как рак, раздувает Митька-форейтор; на дворе сыро и туманно, как будто пар подымается от пахучего навоза; солнышко веселым, ярким светом освещает восточную часть неба, и соломенные крыши просторных навесов, окружающих двор, глянцевины от росы, покрывающей их.

Лев Толстой

### ПАРУС

Белеет парус одинокой  
В тумане моря голубом!..  
Что ищет он в стране далекой?  
Что кинул он в краю родном?..

Играют волны — ветер свищет,  
И мачта гнется и скрипит...  
Увы! Он счастья не ищет  
И не от счастья бежит!

Под ним струя светлей лазури,  
Над ним луч солнца золотой...  
А он, мятежный, просит бури,  
Как будто в бурях есть покой!

*Михаил Лермонтов*

### ПРИКЛЮЧЕНИЯ АЛИСЫ В СТРАНЕ ЧУДЕС

Алиса отворила дверцу и увидела, что она ведет в узкий проход величиной с крысину норку. Она встала на колени и, взглянув в глубину прохода, увидела в круглом просвете уголок чудеснейшего сада. Как потянуло ее туда из сумрачной залы, как захотелось ей там побродить между высоких нежных цветов и прохладных светлых фонтанов!

Случилось столько необычайного за последнее время, что Алисе уже казалось, что на свете очень мало действительно невозможных вещей.

Постояла она у дверцы, потопталась, да и вернулась к столику, смутно надеясь, что найдет на нем какую-нибудь книжку правил для людей, желающих складываться по примеру подозрительной трубы. На этот раз она увидела на нем скляночку, и на бумажном ярлычке, привязанном к горлышку, были напечатаны красиво и крупно два слова: «ВЫПЕЙ МЕНЯ».

Очень легко сказать: «Выпей меня», но умная Алиса не собиралась действовать опрометчиво. «Посмотрю сперва, — сказала она, — есть ли на ней пометка „Яд“».

*По Льюису Кэрроллу*

- ⑩ **УПРАЖНЯЙТЕСЬ!** Отредактируйте тексты, предложенные учителем. Используйте при редактировании команды меню **Home** и кнопки в закладках стандартных инструментов.



## 1.2. Форматирование символов

*Ключевые термины:*

- техноредактирование
- шрифт
- стиль отображения
- форматирование символов

Письменность была изобретена около 5000 лет назад и до сих пор является важнейшим средством хранения и передачи информации. Развитие человеческой цивилизации неразрывно связано с такими шагами развития техники письма, как рукопись, изобретение книгопечатания, компьютерная обработка текстов.

**Техническая и графическая подготовка рукописи к печати называется техноредактированием.**

Техноредактирование реализуется на трех уровнях — страницы, абзаца, символа — и предназначено для создания аккуратно оформленных и легко читаемых текстов. На уровне символа пользователь задает или изменяет свойства каждого символа отдельно или в составе фрагмента текста. В приложении **Word** основными свойствами любого символа являются:

- шрифт;
- стиль отображения;
- размер;
- цвет;
- специальные эффекты.

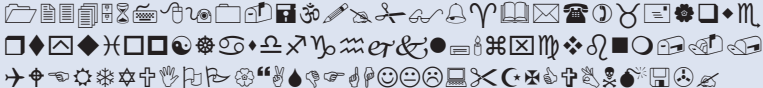
**Шрифт представляет собой полный набор символов (прописных и строчных букв алфавита, цифр, знаков препинания, знаков арифметических операций), имеющих единое графическое оформление.**

В операционной системе **Windows** шрифты хранятся в файлах с расширением **.ttf** и используются во всех приложениях, где предусмотрена обработка текста. Эти файлы разрабатываются художниками, программистами и другими специалистами издательского дела. У каждого шрифта есть собственное название. Все шрифты защищены авторскими правами. В настоящее время существует более 1500 шрифтов. Описание некоторых из них приведено в *табл. 1.1*.

Таблица 1.1

Часто используемые шрифты

Название шрифта	Образец
Arial	Содержит буквы с правильными пропорциями и четкой прорисовкой. Используется для заголовков.

Название шрифта	Образец
Courier Courier New	Имитирует вид символов, используемых в механических пишущих машинках. Все символы одинаковой ширины. Применяется в таблицах, коммерческой корреспонденции, отчетах.
Times New Roman	Символы с малыми засечками (serif) как бы направляют взгляд от одной буквы к другой. Засечки улучшают читаемость слов и шрифта. Используется для больших текстов: учебников, литературных произведений, публикаций.
Wingdings (Звук крыла)	Разнообразные символы и графические знаки, используемые в качестве орнамента или для специальных обозначений: 

**Стиль отображения** задает внешний вид символам, выводимым на экран или отпечатанным на бумаге. Приложение **Word** предлагает следующие стили отображения (см. табл. 1.2):

**Regular** (Обычный) — символы отображаются в точности такими, как и были разработаны создателями шрифта;

**Italic** (Курсив) — символы наклонены на 15°;

**Bold** (Полужирный) — толщина линий, которыми нарисованы символы, увеличена;

**Bold Italic** (Полужирный курсив) — позволяет одновременно применять стили **Bold** и **Italic**.

Таблица 1.2

Стили отображения

Стиль	Образец			
Regular	Arial	Bookman	Courier	Times New Roman
<i>Italic</i>	<i>Arial</i>	<i>Bookman</i>	<i>Courier</i>	<i>Times New Roman</i>
<b>Bold</b>	<b>Arial</b>	<b>Bookman</b>	<b>Courier</b>	<b>Times New Roman</b>
<b><i>Bold Italic</i></b>	<b><i>Arial</i></b>	<b><i>Bookman</i></b>	<b><i>Courier</i></b>	<b><i>Times New Roman</i></b>

**Размеры символов** измеряются в типографских **пунктах**. Один пункт равен 0,351 мм. В качестве примера в табл. 1.3 представлены размеры шрифтов, наиболее часто используемых в учебниках, литературных произведениях, газетах и деловой переписке.

Таблица 1.3

Размеры символов

Размер, в пунктах	Образец			
8	Arial	Bookman	Courier	Times New Roman

10	Arial	Bookman	Courier	Times New Roman
12	Arial	Bookman	Courier	Times New Roman
14	Arial	Bookman	<b>Courier</b>	Times New Roman
16	Arial	Bookman	<b>Courier</b>	Times New Roman

Символы произвольного текста могут быть выведены на экран или отпечатаны на бумаге различными цветами. Каждому символу можно также придать специальные эффекты. Некоторые из эффектов можно увидеть только на экране (например, анимация в стиле *Огни Лас-Вегаса*), другие будут видны и в отпечатанном виде (например, зачеркивание, верхние и нижние индексы).

**Процесс задания свойств символов обрабатываемого текста (шрифт, стиль отображения, размер, цвет, наличие или отсутствие специальных эффектов) называется форматированием символов.**

Форматирование символов (букв, цифр, знаков препинания и т. п.) осуществляется с помощью команды **Font** в закладке инструментов меню **Home**, при активизации которой появляется диалоговое окно (рис. 1.3).

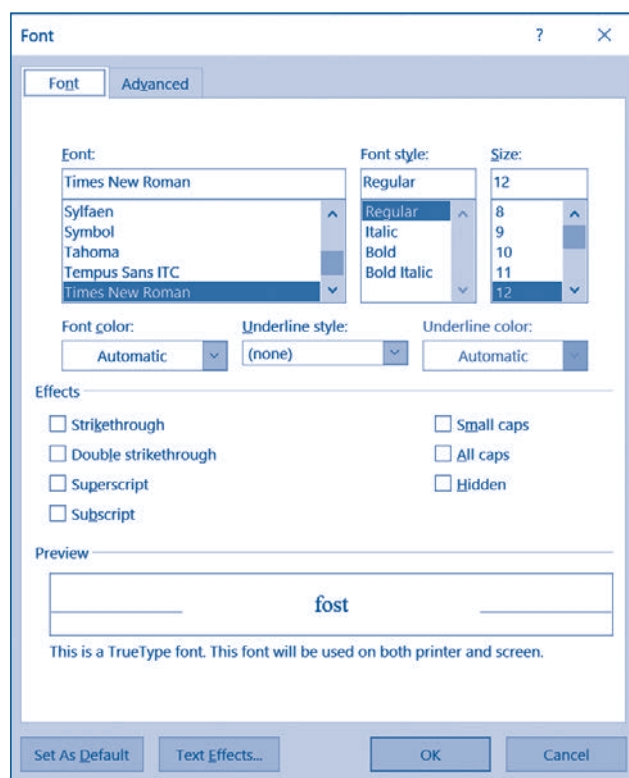


Рис. 1.3. Диалоговое окно **Font**

Установки, заданные в этом окне, будут применены к выделенной в данный момент части текста или, если текст не выделен, к тому слову, внутри которого находится точка вставки. Если же точка вставки находится вне слова, то все установки, выполненные в диалоговом окне **Font**, будут применены к тесту, который пользователь будет вводить с клавиатуры.

Быстрый доступ к некоторым настройкам диалогового окна **Font** возможен с помощью закладки инструментов **Font** в меню **Home**. В закладке содержатся раскрывающиеся списки **Font**, **Font Size** (Размер шрифта), **Font color** (Цвет шрифта) и кнопки **Bold**, **Italic**, **Underline** (Подчеркивание), **Highlight** (Выделение). Управляющие элементы закладки форматирования служат также для отображения текущих свойств обрабатываемых символов. Они меняют свое состояние в зависимости от свойств символов и абзацев, в которых в данный момент находится точка вставки.

## Вопросы и упражнения

- 1 Объясните термин *техноредактирование*. Когда и каким образом оно осуществляется?
- 2 **ИССЛЕДУЙТЕ!** Назовите основные свойства символов. Как в случае приложения **Word** можно определить эти свойства?
- 3 Объясните термин *шрифт*. Каким образом хранятся шрифты в операционной системе **Windows**?
- 4 Найдите на жестком диске папку **Fonts**. Выведите на экран и отпечатайте наиболее часто используемые шрифты.
- 5 Назовите стили отображения, используемые в приложении **Word**.
- 6 Определите, какими стилями изображены символы в следующих словах:

<u>школа</u>	класс	учебник
автомобиль	компьютер	дерево

Введите в компьютер и отформатируйте слова по приведенной выше модели. Заметьте, как меняется состояние кнопок **Bold**, **Italic** и **Underline** при переводе точки вставки с одного слова на другое.

- 7 Объясните термин *форматирование*. Как выполняется форматирование символов в приложении **Word**?
- 8 **ИССЛЕДУЙТЕ!** Укажите все управляющие элементы, предназначенные для форматирования символов, находящиеся на закладке форматирования (**Font**). Объясните, как ими пользоваться.
- 9 **ЭКСПЕРИМЕНТИРУЙТЕ!** Введите в компьютер и отформатируйте текст по приведенному ниже образцу:

Arial	Arial	Arial	Arial
Courier	Courier	Courier	Courier
Bookman	Bookman	Bookman	Bookman
Times	Times	Times	Times

Используйте символы размером 12, 14, 16 и 18 пунктов. Заметьте, как меняется состояние элементов управления панели форматирования при переводе точки вставки с одного слова на другое.

- ⑩ **ИССЛЕДУЙТЕ!** Отредактируйте в приложении **Word** тексты:

### **ЗВЕЗДА ПОЛЕЙ**

Звезда полей во мгле заледенелой,  
Остановившись, смотрит в полынью.  
Уж на часах двенадцать прозвенело,  
И сон окутал родину мою.

Звезда полей! В минуты потрясений  
Я вспоминал, как тихо за холмом  
Она горит над золотом осенним,  
Она горит над зимним серебром...

Звезда полей горит, не угасая,  
Для всех тревожных жителей земли.  
Своим лучом приветливым касаясь  
Всех городов, поднявшихся вдали.

Но только здесь, во мгле заледенелой,  
Она восходит ярче и полней,  
И счастлив я, пока на свете белом  
Горит, горит звезда моих полей...

*Николай Рубцов*

### **ГЛОССА**

То, что было, то и будет,  
Всё старо и ново в жизни.  
Что тебе в добре и хude —  
Сам сочти и поразмысли.

Страх и чаянья напрасны:  
Жизнь — волна в пустыне водной.  
Ни призыву, ни соблазну  
Не внемли душой холодной.

Много зрелищ взор тревожит,  
Много звуков слух терзает.  
Кто постичь всё это может,  
Кто взирает и внимает?

Сам себе служи законом,  
Пусть себя лишь сердце судит.  
В этом мире пустозвонном  
То, что было, то и будет.

*Михай Эминеску  
(перевод Ю. Кожевникова)*

Сохраните их под именами **Zvezda**, **Glossa** соответственно.

- ① **УПРАЖНЯЙТЕСЬ!** Загрузите в приложение **Word** тексты **Otrocestvo** и **Alisa**. Отформатируйте символы этих текстов следующим образом:
- **заголовок:** Arial, 14 пунктов, Bold;
  - **основной текст:** Times New Roman, 12 пунктов, Regular;
  - **автор:** Courier New, 10 пунктов, Italic.
- Сохраните отредактированные документы на диске.
- ② **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Измените шрифт **Times New Roman** в документах **Alisa**, **Parus** и **Otrocestvo** на шрифт **Courier New**. Какие из текстов легче читаются? Как изменяется площадь, занимаемая текстом на странице, при смене шрифта?
- ③ **СОЗДАЙТЕ!** Попытайтесь найти наиболее удачное, с вашей точки зрения, форматирование символов в файлах **Zvezda**, **Glossa**, **Parus**. Назовите преимущества и недостатки рассмотренных вариантов.

### 1.3. Форматирование абзацев

*Ключевые термины:*

- абзац
- выравнивание
- отступы
- интервалы

Для лучшего зрительного восприятия большой текст делится на небольшие фрагменты, называемые **абзацами**. Обычно каждый абзац посвящен определенной теме, а переход к новому абзацу означает смену темы. В приложении **Word** признак окончания абзаца вводится нажатием клавиши **<Enter>**. На экране признак конца абзаца обозначается символом **¶**. Абзацы имеют следующие свойства:

- выравнивание;
- отступы;
- отступ первой строки (так называемая красная строка);
- интервал между строками внутри абзаца;
- интервал до и после абзаца;
- специальные эффекты на уровне абзаца.

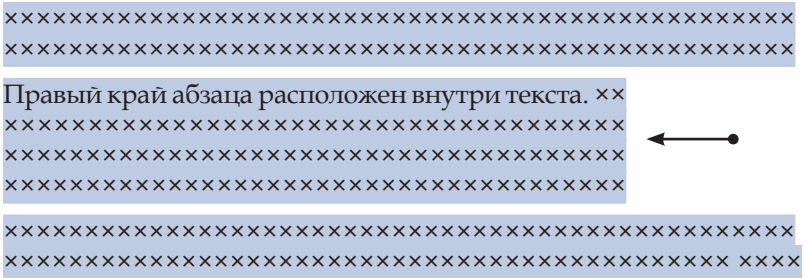
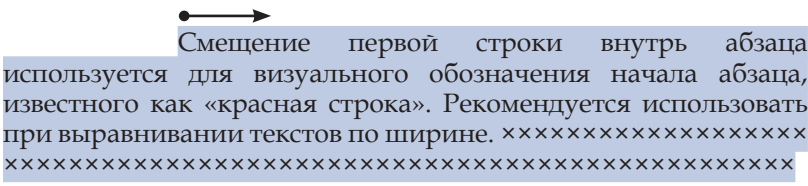
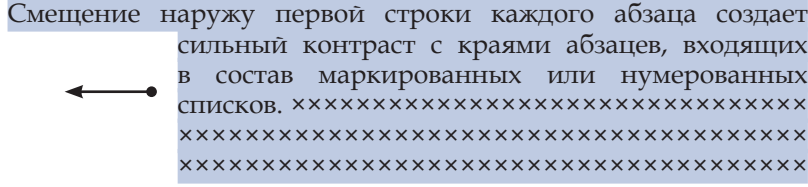
**Выравнивание** определяет, каким образом располагаются начало и конец каждой строки абзаца относительно левого или правого края, центра или обоих краев одновременно (табл. 1.4).

Таблица 1.4

**Выравнивание абзацев**

<b>Left</b> (Влево)	Каждая строка абзаца начинается на одинаковом расстоянии от левого края и заканчивается, в зависимости от количества символов в соответствующей строке, на разных расстояниях от правого края.
------------------------	--



<b>Right</b> (Правый отступ)	 <p>Правый край абзаца расположен внутри текста. xx</p>
<b>Special</b> (Специальный отступ)	<p>Начало первой строки (<b>First line</b>) смещено внутрь или наружу абзаца.</p>  <p>Смещение первой строки внутрь абзаца используется для визуального обозначения начала абзаца, известного как «красная строка». Рекомендуется использовать при выравнивании текстов по ширине.</p>  <p>Смещение наружу первой строки каждого абзаца создает сильный контраст с краями абзацев, входящих в состав маркированных или нумерованных списков.</p>

Расстояние между строками (**Line space**) каждого абзаца устанавливается автоматически в зависимости от размера используемых символов. Однако с некоторыми документами легче работать, если межстрочное расстояние установлено чуть больше. Например, дополнительное расстояние может понадобиться, если кто-то хочет внести вручную исправления в соответствующий абзац.

Расстояние до (**Before**) и после (**After**) абзаца задает дополнительное пространство, которое добавляется перед первой и соответственно после последней строки абзаца. Свободное пространство позволяет пользователю быстро находить начало каждого абзаца, делая текст легко читаемым. Дополнительное расстояние рекомендуется вводить для текстов, выровненных по левому краю.

**Процесс задания свойств абзацев обрабатываемого текста (выравнивание, отступ, межстрочное расстояние, наличие или отсутствие спецэффектов) называется форматированием абзацев.**

Форматирование абзацев осуществляется с помощью команд **Paragraph** (Абзац) и **Borders and Shading** (Границы и Заливка) из меню **Format**. При выборе пункта меню **Paragraph** появляется диалоговое окно (рис. 1.4), в которое вводятся соответствующие установки.

Введенные установки будут применены к выбранным абзацам или, при отсутствии выбора, к абзацу, внутри которого расположена точка вставки. Если



точка вставки в начале нового абзаца, то установки диалогового окна будут применены к тексту, который вводится с клавиатуры.

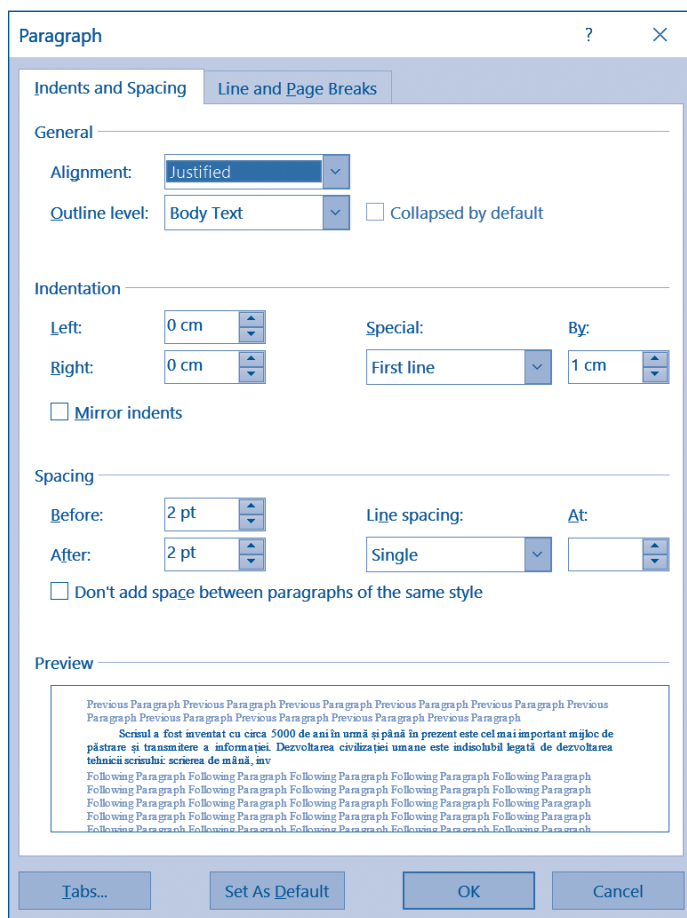


Рис. 1.4. Диалоговое окно **Paragraph** (Абзац)

Быстрый доступ к элементам управления диалогового окна **Paragraph** осуществляется с помощью кнопки **Paragraph Settings** инструментов закладки форматирования и горизонтальной линейки. Панель инструментов форматирования содержит кнопки **Align Left**, **Center**, **Align Right**, **Justify**, предназначенные для выравнивания текста. Горизонтальная линейка содержит указатели **First Line Indent** (↵), **Left Indent** (⏏) и **Right Indent** (⏏), предназначенные для настройки горизонтальных отступов абзацев. Напоминаем, что элементы управления на закладке инструментов и горизонтальной линейке используются и как индикаторы свойств соответствующих абзацев.

При выборе из меню **Format** пункта **Borders and Shading** на экран выводится диалоговое окно (рис. 1.5), которое позволяет задавать ряд эффектов на уровне абзаца: обрамлять абзацы рамками различных типов, закрашивать абзацы (размещать текст на окрашенном фоне) и т. п.

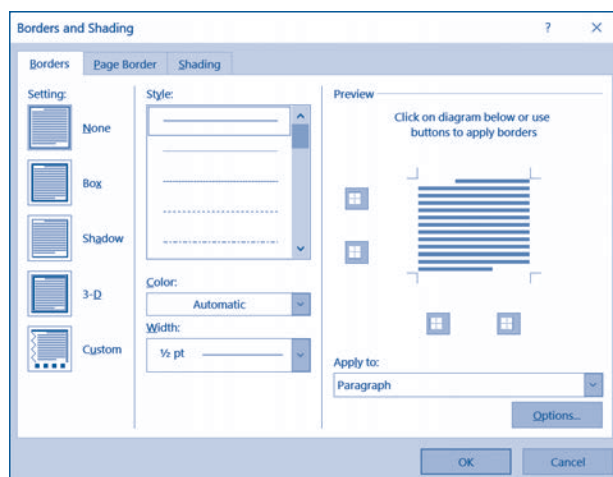


Рис. 1.5. Диалоговое окно **Borders and Shading**

## Вопросы и упражнения

- ❶ С какой целью текст делится на абзацы? Приведите примеры.
- ❷ Как отделяются друг от друга абзацы в приложениях **Notepad** и **Word**?
- ❸ **ИССЛЕДУЙТЕ!** Назовите основные свойства абзаца. Как можно узнать эти свойства?
- ❹ Как можно выравнивать строки абзацев? В каких случаях используют способы выравнивания?
- ❺ Какие типы отступа для абзацев вы знаете? Опишите, в каких случаях применяют каждый из типов отступа.
- ❻ **ЭКСПЕРИМЕНТИРУЙТЕ!** Как установить интервал между строками внутри абзаца и между абзацами?
- ❼ Объясните термин *форматирование абзацев*.
- ❽ **ИССЛЕДУЙТЕ!** Используя справочную систему **Help**, найдите назначение всех элементов управления диалогового окна **Paragraph**. Заметьте, как меняется вид текста в области предварительного просмотра этого окна.
- ❾ Укажите на горизонтальной линейке и на закладке инструментов форматирования элементы управления, предназначенные для форматирования абзацев. Объясните, как ими пользоваться.
- ❿ **ИССЛЕДУЙТЕ!** Используя справочную систему **Help**, найдите назначение всех элементов управления диалогового окна **Borders and Shading** из выпадающего списка **Borders**. Заметьте, как изменяется вид текста в области предварительного просмотра этого окна.
- ⓫ Откройте в приложении **Word** тексты **Alisa** и **Otrocestvo**. Отформатируйте абзацы этих текстов по образцу, показанному на страницах 7 и 8 данного учебника. Проверьте параметры форматирования с помощью команды "↗?" справочной системы.
- ⓫ Отформатируйте документы **Parus**, **Zvezda** по образцу, показанному на страницах 8 и 13 данного учебника. Выведите на экран свойства символов и абзацев отформатированных документов.

## 1.4. Форматирование страниц

Ключевые термины:

- физическая страница
- логическая страница
- колонтитулы
- раздел

При редактировании документа приложение **Word** делит его на страницы. Когда страница заполнена, приложение автоматически начинает новую страницу. Чтобы принудительно начать новую страницу до того, как закончена текущая, применяется **разделитель страниц (Page break)**.

Каждая страница характеризуется следующими параметрами (рис. 1.6):

- физическими размерами;
- ориентацией;
- отступами;
- размерами областей колонтитулов (верхнего и нижнего);
- выравниванием текста по вертикали.

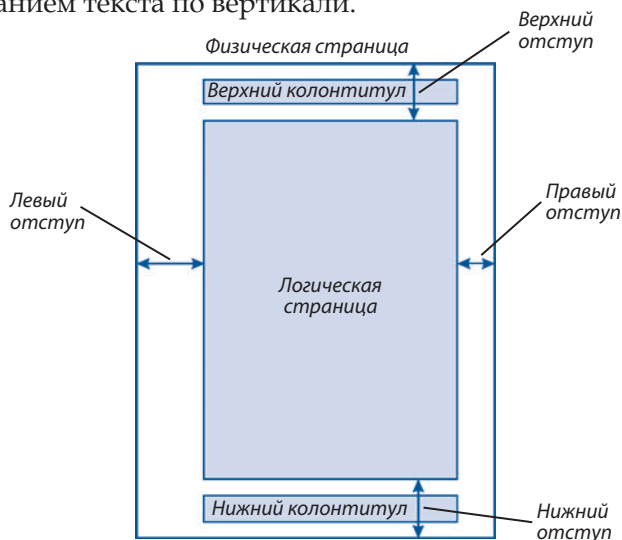


Рис. 1.6. Структура страницы

**Процесс задания свойств страниц (размеров, ориентации, отступов, выравнивания по вертикали, размеров колонтитулов) называется форматированием страниц.**

Страница форматируется с помощью меню **Page Layout**, кнопки **Page Setup** на закладке инструментов, при этом на экран выводится диалоговое окно (рис. 1.7).

Элементы управления рассматриваемого окна имеют следующее назначение:

**Margins** (Отступы) — установка размеров отступов;

**Paper Size** (Размеры листа) — установка размеров физической страницы и ее ориентации;

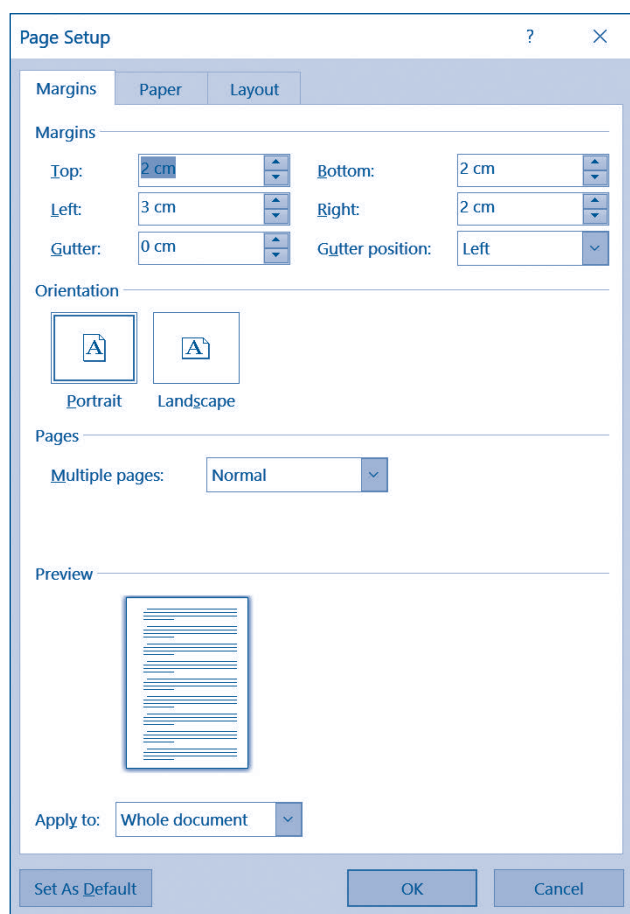


Рис. 1.7. Диалоговое окно **Page Setup**

**Paper Source** (Источник бумаги) — указывает способ подачи бумаги в принтер;  
**Layout** (Раскладка) — устанавливает вертикальное выравнивание текста на странице (если страница целиком заполнена).

Быстрый доступ к некоторым параметрам страницы **Margins** осуществляется с помощью горизонтальной и вертикальной линеек. Светлая часть каждой линейки показывает размеры логической страницы, а затемненная часть — расстояние от границ листа до границ текста. Размеры отступов можно изменить, «перетягивая» границы светлой и затемненной области в нужном направлении.

Пустое пространство вокруг текста создает впечатление элегантного, «дружественного» документа. Одновременно боковые отступы дают читателю возможность держать его в руках, не закрывая при этом текст и изображения.

**Колонтитулы (верхний и нижний)** являются фрагментами текста и повторяются в верхней и/или нижней части каждой страницы документа (рис. 1.6). Верхний колонтитул появляется в верхней части страницы над основным текстом, а нижний колонтитул — в ее нижней части.

Заголовки создаются пользователем с помощью команды **Header** (Верхний колонтитул), а нижние колонтитулы — с помощью команды **Footer** (Нижний

колонтитул) на закладке **Header & Footer**, меню **Insert**. При запуске этих команд на экране появляются списки, содержащие различные модели, из которых пользователь может выбрать наиболее подходящую для его документов (рисунк 1.8).

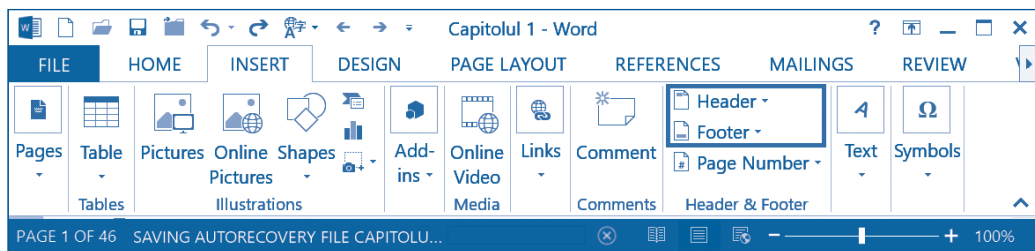


Рис. 1.8. Команды **Header** и **Footer**

Помимо команд **Header** и **Footer** на закладке содержится раскрывающийся список с набором кнопок, которые позволяют выполнить автоматическую вставку в колонтитул имени файла, текущих даты и времени, общего количества страниц документа и другие операции.

Отметим, что форматирование страниц — обширная операция, которая автоматически распространяется на все страницы документа. Вместе с тем, при работе с некоторыми документами может возникнуть ситуация, когда необходимо использовать два или более видов форматирования страниц. Например, школьный учебник содержит титульный лист, основной текст, приложения, список литературы, оглавление/содержание, причем соответствующие страницы отформатированы по-разному. Установка различных способов форматирования страниц осуществляется разбивкой документов на разделы.

**Разделы представляют собой непрерывные области документа, для каждой из которых можно применить различные способы форматирования страниц.**

Для перехода к новому разделу применяются разделители **Section Break**. Вставка разделителей страниц и разделов выполняется с помощью команды **Page Layout, Breaks**, при выполнении которой появляется диалоговое окно (рис. 1.9).

Это окно позволяет вставлять разделители разделов в произвольные места документа: в начало новой страницы (**Next page**), внутрь текущей страницы в то место, где находится курсор (**Continuous**), в начало новой страницы с четным (**Even Page**) или нечетным (**Odd Page**) номером. После вставки нужных разделителей пользователь может отформатировать страницы каждого раздела отдельно. Для автоматической нумерации страниц используется команда **Insert, Page Number** (Вставка, Нумерация).

## Вопросы и упражнения

- ❶ Для чего предназначены разделители страниц? Содержит ли данный учебник такие разделители?

② Назовите основные свойства страниц документа.

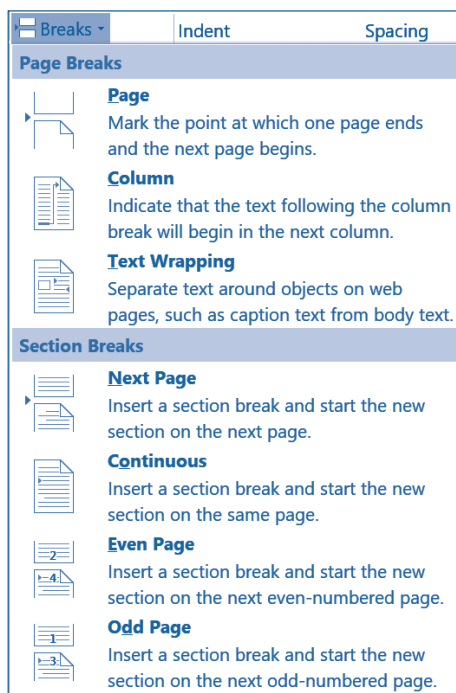


Рис. 1.9. Диалоговое окно **Breaks**

- ③ **ИССЛЕДУЙТЕ!** Определите следующие свойства страниц данного учебника: физические размеры, ориентацию, размеры отступов, выравнивание текста по вертикали.
- ④ Для чего предназначены разделители разделов? Содержит ли данный учебник такие разделители? Ответ обоснуйте.
- ⑤ **ИССЛЕДУЙТЕ!** Найдите на горизонтальной и вертикальной линейках элементы управления, предназначенные для форматирования страниц.
- ⑥ **ИССЛЕДУЙТЕ!** Используя справочную систему, найдите назначение всех опций, предлагаемых диалоговыми окнами **Page Setup**, **Breaks** и списками инструментов **Header** и **Footer**.
- ⑦ Для чего предназначены колонтитулы? Какую информацию может содержать колонтитул?
- ⑧ **УПРАЖНЯЙТЕСЬ!** Отформатируйте страницы ранее созданных документов по образцам, предложенным преподавателем. Верхний колонтитул каждой страницы должен содержать в левой части текущую дату и время, а нижний колонтитул — название файла, имя и фамилию ученика.
- ⑨ **СОЗДАЙТЕ!** Создайте документ, который содержит все ранее отредактированные вами документы: **Alisa**, **Glossa**, **Parus**, **Otrocestvo**, **Zvezda**. Новый документ должен сохранить все параметры форматирования страниц, абзацев и символов исходных документов. Пронумеруйте страницы созданного документа.
- ⑩ Объясните термин *форматирование страниц* и укажите элементы управления, предназначенные для этой операции.
- ⑪ **ИССЛЕДУЙТЕ!** Может ли один и тот же документ содержать страницы с *книжной* и *альбомной* ориентацией?

- ❷ **ЭКСПЕРИМЕНТИРУЙТЕ!** Для облегчения процесса чтения текст документа можно разместить на странице в форме столбцов или колонок (*Columns*). Для этого выделите нужную часть текста и выполните команду **Page Layout, Columns**. Скопируйте файлы **Alisa, Otrocestvo** в единый документ и разбейте текст на столбцы.

## 1.5. Списки и таблицы

*Ключевые термины:*

- список
- элемент списка
- таблица
- ячейка

Списки предназначены для того, чтобы зрительно выделить тематически связанные абзацы текста. Например, в конце каждого параграфа настоящего учебника можно найти список, состоящий из вопросов и упражнений.

**Список представляет собой фрагмент текста, в котором начало каждого абзаца выделено с помощью специального символа или порядкового номера. Соответствующие абзацы называются элементами списка.**

Для создания списка с маркерами выделяются нужные абзацы и выполняется команда **Home, Bullets**. После ее активизации приложение выводит на экран диалоговое окно **Bullet Library** (Библиотека маркеров) (рис. 1.10), где можно выбрать вид маркеров, с помощью которых будут отмечены элементы списка.

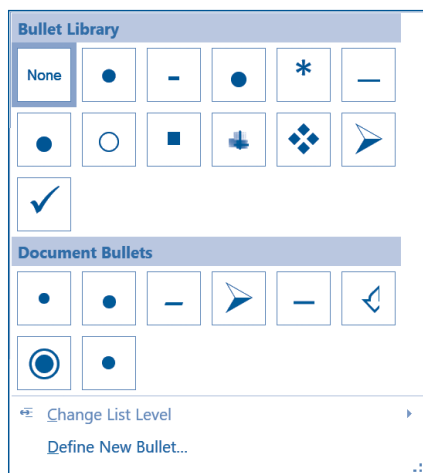


Рис. 1.10. Диалоговое окно **Bullet Library**

Если необходимо использовать другие маркеры, следует нажать кнопку **Define New Bullet** (Определение нового маркера). При этом на экран выводится

окно, в котором пользователь может выбрать самые разнообразные знаки выделения, такие как значки, пиктограммы, символы и т. д.

Аналогичным образом создаются нумерованные списки. При запуске команды **Home, Numbering** на экране отображается диалоговое окно **Numbering Library**, которое содержит различные параметры для форматирования списков, которые создаются или формируются (рисунок 1.11).

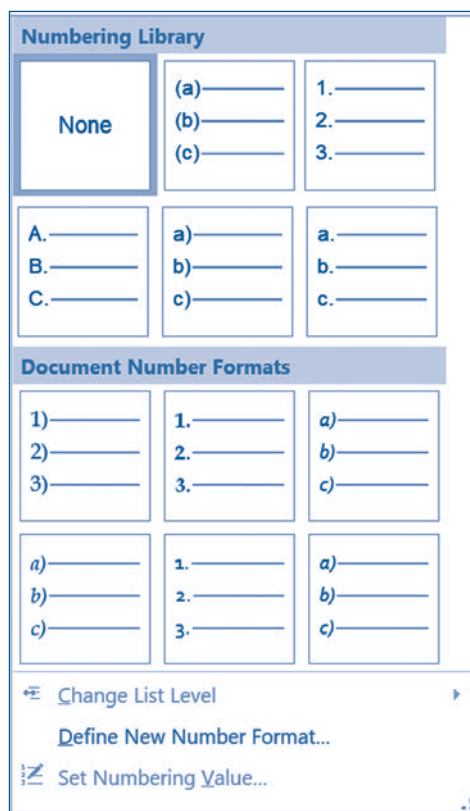


Рис. 1.11. Диалоговое окно **Numbering Library**

Чтобы представить в удобном для чтения виде большой объем однотипных данных, используются таблицы.

**Таблица** представляет собой составной объект, состоящий из строк и столбцов. Прямоугольники, образованные пересечением строк и столбцов, называются ячейками.

Каждая ячейка не зависит от остальных ячеек и может содержать:

- текст;
- числа;
- изображения;
- формулы;
- объекты, созданные другими приложениями.

Структура таблиц приложения **Word** представлена на рис. 1.12.



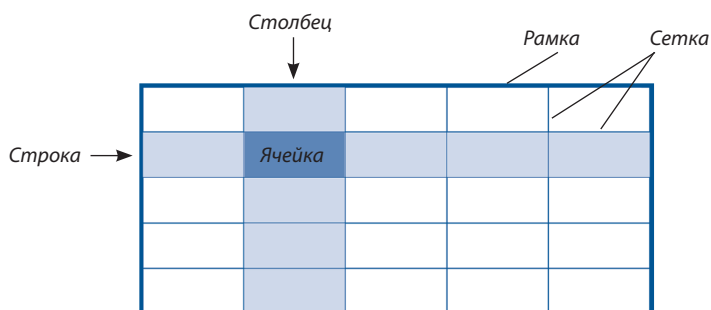


Рис. 1.12. Структура таблиц

Самый простой способ создания таблиц заключается в активизации команды **Table, Insert Table** из меню **Insert**. При этом на экране появляется диалоговое окно (рис. 1.13), элементы управления которого позволяют задавать нужное число строк (**Number of rows**) и столбцов (**Number of columns**). Создав таблицу, пользователь может редактировать информацию в ее ячейках.

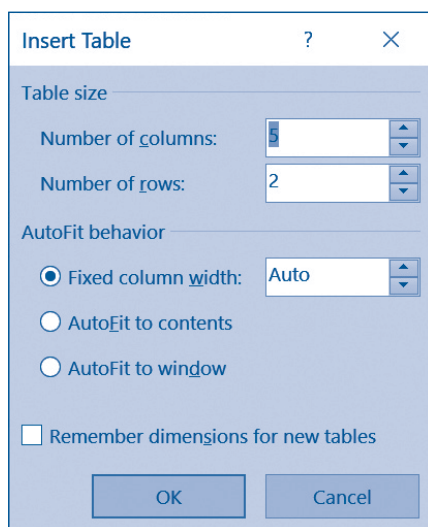


Рис. 1.13. Диалоговое окно **Insert Table**

С таблицей можно выполнять следующие **операции**:

- стирание;
- копирование;
- разбивка таблицы на две части;
- изменение размеров ячеек, строк или столбцов;
- добавление строк или столбцов;
- слияние двух ячеек;
- разбивка любой ячейки на две и более ячеек.

Большинство команд, с помощью которых можно выполнить указанные операции, находятся в меню **Design** и **Layout**, которые появляются на ленте меню при редактировании таблицы. (рис. 1.14)

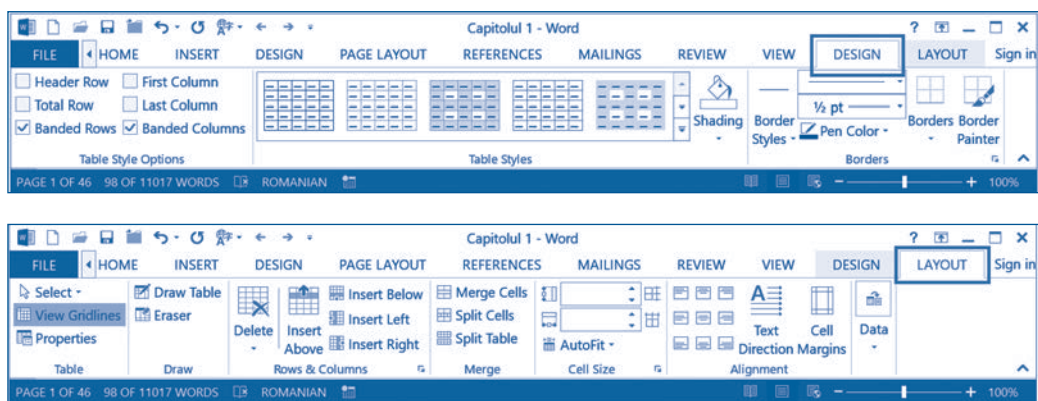


Рис. 1.14. Меню **Design** и **Layout**

Быстрый доступ к некоторым командам обработки таблиц достигается с помощью кнопок на закладках инструментов форматирования или в контекстных меню. Эти меню содержат команды (инструменты) для рисования сетки и выравнивания содержимого каждой ячейки: карандаш, ластик, цветовая заливка, шаблоны линий, вертикальное выравнивание и т. д. Обратите внимание, что эти инструменты также можно использовать для изменения ранее созданных таблиц.

Вид создаваемых таблиц зависит в значительной степени от способа распределения данных по отдельным ячейкам и их формата. При создании и форматировании таблиц должны соблюдаться следующие правила:

1. Строки и столбцы таблицы должны отделяться друг от друга линиями сетки или цветами заливки.
2. Для заголовков строк и столбцов должны использоваться форматы, явно отличающие их от остальных ячеек.
3. Нужно избегать слишком близкого расположения текста и линий сетки, а также текстов, находящихся в соседних столбцах.

Отметим, что использование таблиц, сетки и границы которых невидимы, позволяет упорядочить размещение чисел, текстов и изображений, а также создавать документы, содержащие в одной строке требуемое число абзацев.

## Вопросы и упражнения

- ❶ Объясните термины *список* и *элемент списка*. Для чего применяют списки?
- ❷ **ИССЛЕДУЙТЕ!** Найдите в данном учебнике фрагменты текста, представляющие собой списки. Каким образом выделяют каждый элемент списка?
- ❸ **СОЗДАЙТЕ!** Создайте документ, содержащий список учащихся вашего класса.
- ❹ **УПРАЖНЯЙТЕСЬ!** Наберите и отпечатайте на принтере следующий список.

*Ключевые термины:*

- документ;
- структура документа;
- окно документа.

- ❺ Для чего предназначены таблицы? Назовите составные части таблицы.

- ⑥ **ИССЛЕДУЙТЕ!** Определите количество строк, столбцов и ячеек в *табл. 1.1, 1.2 и 1.4* настоящей главы. Опишите, чем отличается формат заголовков столбцов от формата остальных ячеек.
- ⑦ Наберите и распечатайте на принтере *табл. 1.2 и 1.3* настоящего учебника.
- ⑧ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Каковы преимущества и недостатки двух методов создания таблиц: вставки и черчения?
- ⑨ **ИССЛЕДУЙТЕ!** Найдите в меню **Design** и **Layout**, в закладках инструментов, кнопки создания и форматирования таблиц. Поясните, как ими пользоваться.
- ⑩ **ИССЛЕДУЙТЕ!** Пользуясь справочной системой, найдите назначение всех команд меню **Design** и **Layout**.
- ⑪ Выведите на экран стили форматирования таблиц, которые содержатся в диалоговом окне команды **Table Styles**, меню **Design**.
- ⑫ **СОЗДАЙТЕ!** Используя таблицу, сетка и границы которой невидимы, создайте следующий документ:

СПИСОК участников олимпиады по информатике			
1.	Мунтяну Ион	Орхейский район, село Киперчень	тел. 23-619
2.	Петреску Ангела	г. Бухарест, Каля Доробанцилор 3, кв. 15	тел. 261-32-64
3.	Кирияк Петру	Сынжерейский район, село Копэчень	тел. 16-215
4.	Маслов Владимир	г. Кишинэу, Куза-Водэ 8, кв. 6	тел. 32-65-84

- ⑬ **УЧИТЕСЬ УЧИТЬСЯ!** В дополнение к спискам, рассмотренным в этом параграфе, приложение **Word** предлагает пользователю возможность создавать многоуровневые списки (**Multilevel List**). Используя справочную систему, узнайте, как создавать такие списки.
- Создайте список, содержащий на первом уровне — марки автомобилей, а на втором — некоторые модели автомобилей этой марки. Например, *Dacia* — это марка автомобилей, а *Dokker*, *Duster*, *Logan*, *Lodgy*, *Sandero* — модели этой марки. Другие марки автомобилей — *Ford*, *Mazda*, *Toyota*, *Volkswagen* и т. д.

## 1.6. Вставка объектов

*Ключевые термины:*

- вставка из буферной памяти
- вставка из других приложений
- вставка из другого файла

Как известно, создаваемые документы могут содержать разнообразные объекты: таблицы, изображения, звуковые и видеофрагменты. В зависимости

от способа создания объектов, которые можно включать в документ, различают:

- 1) объекты, созданные внутри приложения **Word**;
- 2) объекты, созданные с помощью других приложений.

Примером объектов первого типа служат таблицы, созданные при помощи команд из меню **Insert, Table**. В качестве примеров объектов второго типа можно привести рисунки, созданные в приложении **Paint**, звуковые фрагменты, записанные с помощью приложения **Voice Recorder**.

Операционная система **Windows** предоставляет следующие возможности обмена объектами между приложениями:

- с использованием буферной памяти (**Clipboard**);
- обращением к приложению, в котором будет создан нужный объект;
- с использованием файлов.

Вставка объектов через буферную память предполагает, что одновременно должно быть запущено приложение **Word** и какое-нибудь другое приложение, например **Calculator, Paint, Voice Recorder**. Для вставки объекта в текстовый документ необходимо выполнить следующие операции:

- переключиться в приложение, в котором должен быть создан нужный объект, например в приложение **Paint**;
- скопировать в буферную память нужный объект или его часть (команда **Home, Copy**);
- вернуться в приложение **Word**;
- вставить объект из буферной памяти в документ (команда **Home, Paste**).

Вставку через буферную память целесообразна, если нужна только часть исходного объекта. Например, из рисунка, созданного в приложении **Paint**, можно взять только определенные фрагменты: небо, солнце, дом на горизонте, автомобиль и др.

Вставка из других приложений используют, когда объект, который необходимо вставить в документ, еще не создан. Для вызова приложения, в котором будет создан нужный объект, выполняют команду **Insert, Object** (рис. 1.15).

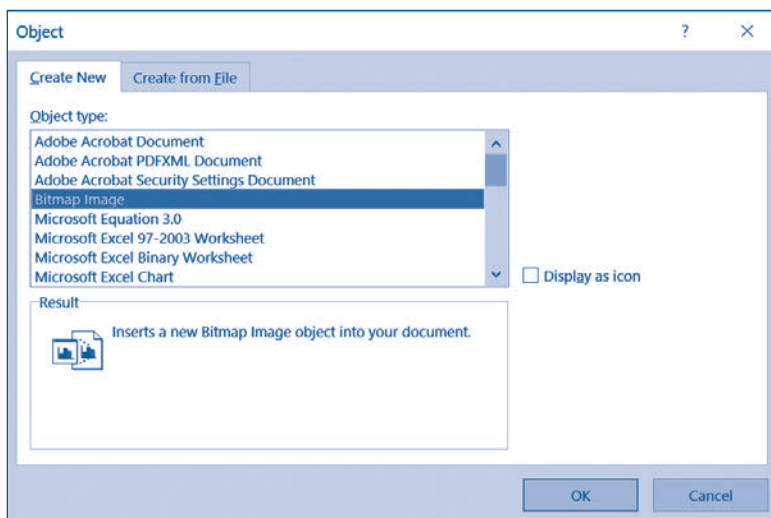


Рис. 1.15. Диалоговое окно **Object**

Страница **Create New** (Создание нового объекта) диалогового окна содержит список **Object Type** (Тип объекта), в котором пользователь должен указать тип создаваемого объекта. В зависимости от выбранного типа автоматически запускается нужное приложение. Например, при выборе типа **Bitmap Image** (Изображение в формате битовая карта) будет запущено приложение **Paint**. Пользователь сможет работать с этим приложением, не покидая **Word**. После записи и возможного редактирования изображения выполняется возврат в приложение **Word** (рис. 1.16).

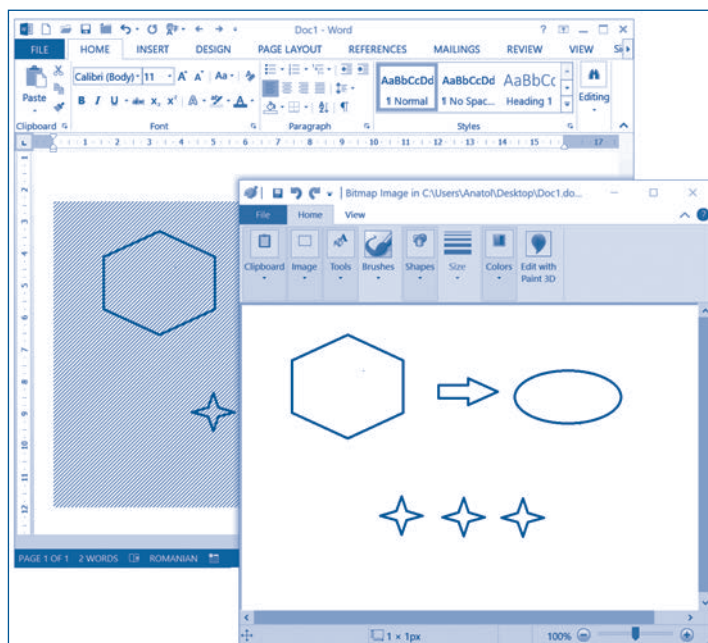


Рис. 1.16. Вставка изображений, созданных в приложении **Paint**

Как видно из рис. 1.15, список **Object Type** содержит большое число различных типов объектов, которые можно включать в документы:

**Adobe Acrobat Document** — переносимые документы;

**Bitmap Image** — изображения в формате битовой карты;

**Microsoft Equation** — формулы и уравнения;

**Microsoft PowerPoint Slide** — диапозитивы;

**Microsoft Graph Chart** — графики и диаграммы.

Создание объектов с помощью соответствующих приложений требует предварительного обучения работе с этими приложениями. Естественно, каждый пользователь изучает только те приложения, которые он непосредственно использует в своей профессиональной деятельности. Например, художники изучают только программы для работы с рисунками, а математики — программы, предназначенные для вставки и редактирования формул, графиков и диаграмм.

**Вставка из другого файла** применяется в том случае, когда соответствующие объекты созданы ранее. Для вставки таких объектов служит страница **Create from File** окна **Object** (рис. 1.17).

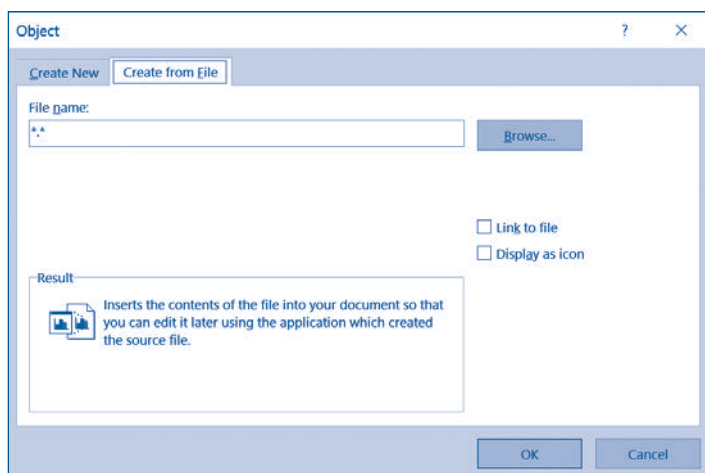


Рис. 1.17. Страница **Create from File**

На этой странице указаны следующие элементы управления:

**Browse** (Обзор). Позволяет выбрать файл, содержащий нужный объект.

**Link to file** (Ссылка на файл). Устанавливает связь между файлом и документом.

Если этот флажок не установлен, выбранный файл становится частью документа **Word**. Когда документ отображается на экране, вставленный файл обозначается соответствующим значком/пиктограммой. Если щелкнуть по этому значку, текстовый редактор автоматически запускает приложение, связанное с данным расширением файла.

Если установлен флажок **Link to file** (Ссылка на файл), сам файл не вставляется в документ **Word**, а только ссылка на него. При изменении файла автоматически будет изменен объект, вставленный в документ. Данную опцию целесообразно применять в случаях, когда в разработке сложного документа участвуют несколько пользователей, каждый из которых отвечает за создание определенных объектов. Например, при создании школьных учебников авторы текстов и художники работают одновременно. При этом художники могут обрабатывать рисунки без прямого вмешательства в документы, с которыми работают авторы.

**Display as icon** (В виде значка). Вставляемый объект будет отображен в документе в виде пиктограммы. Используется только при работе с текстом документа с целью экономии места на экране и времени при просмотре больших документов. Например, на рис. 1.18 показан документ **Word**, в который вставлены три файла.

Первый из файлов, вставленных в документ **Word** в приведенном выше примере, называется "Jules Verne — Capitan la cincisprezece ani" («Жюль Верн — пятнадцатилетний капитан»), представляет собой электронную книгу в формате **.epub**, широко используемом для чтения книг с помощью портативных цифровых устройств.

Второй файл, названный "Ducu Bertzi — Florile dalbe", представляет собой песню на народные мотивы. Этот аудиофайл имеет формат **.mp3**. Файл **.mp3** почти в десять раз меньше исходного аудиофайла, но качество звука почти



такое же, как на оптических дисках. Формат **.mp3** является наиболее популярным форматом хранения музыкальных файлов как на компьютерах, так и на портативных цифровых устройствах.

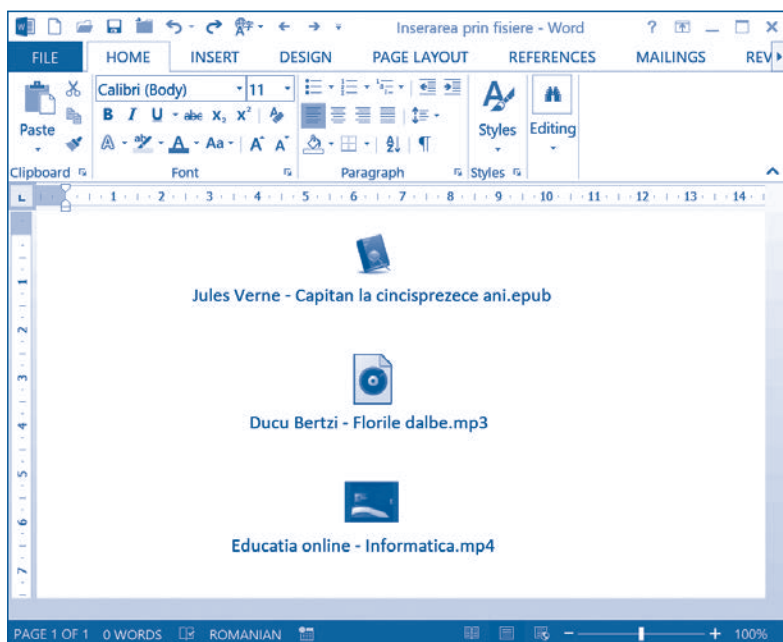


Рис 1.18. Файлы, вставленные в документ **Word**

Файл “Educatia online — informatica” (Онлайн-образование — информатика) имеет формат **.mp4**, который часто используется для хранения видео- и аудиоданных. Этот файл содержит видеоролик урока для гимназистов, обучающихся дистанционно.

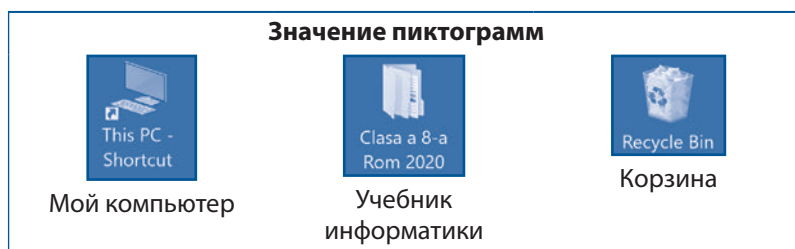
Подчеркнем, что включение мультимедийных файлов в документы **Word** приводят, как правило, к значительному увеличению их размера. Например, в случае видеофильмов это увеличение может достигать порядка гигабайт. Очевидно, что открытие и обработка таких файлов может создать значительные трудности, особенно в случае компьютеров с небольшой производительностью. Безусловно, что передача таких документов по компьютерным сетям займет много времени.

Поэтому в случае мультимедийных файлов наиболее целесообразным решением было бы хранить их на диске персонального компьютера или виртуальном диске, предоставляемом службами веб-хранилища, и вставлять в документы **Word** только ссылки на них.

## Вопросы и упражнения

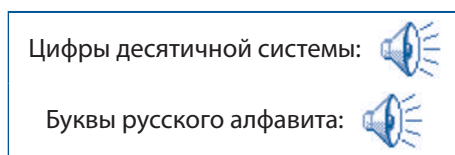
- 1 Как классифицируются объекты, которые можно вставлять в документы?
- 2 Объясните, как вставлять объекты, пользуясь буферной памятью. При каких условиях целесообразно использовать данный метод?

- ③ **СОЗДАЙТЕ!** Пользуясь методом вставки объектов из буферной памяти, создайте следующий документ.



Для размещения текста на странице воспользуйтесь таблицей с невидимой сеткой.

- ④ Объясните метод вставки объектов, основанный на обращении к другим приложениям. При каких условиях целесообразно использовать данный метод?
- ⑤ **СОЗДАЙТЕ!** Пользуясь методом вставки, основанном на обращении к другим приложениям, создайте следующий документ.



Первый аудиофрагмент должен содержать названия цифр десятичной системы: *ноль, один, два, ..., девять*. Второй аудиофрагмент должен содержать названия букв русского алфавита: *а, б, в, ..., я*.

- ⑥ **ИССЛЕДУЙТЕ!** Откройте диалоговые окна команд **Shapes** (Фигуры) и **SmartArt** в меню **Insert**. В каких документах могут быть использованы эти объекты?
- ⑦ Объясните метод вставки объектов из других файлов. При каких условиях целесообразно использовать данный метод?
- ⑧ Объясните назначение всех флажков в диалоговых окнах **Object** и **Create from File** (рис. 1.15 и 1.17).
- ⑨ **РАБОТАЙТЕ В КОМАНДЕ.** Попросите друга создать на другом компьютере изображение, приведенное на рис. 1.16. Скопируйте соответствующий файл по сети и вставьте созданное другом изображение в начало следующего документа:

Эта страница позволяет вставлять в документ объекты из других файлов. При нажатии кнопки **Browse** (Обзор) на экран выводится диалоговое окно с тем же именем.

После текста вставьте в документ диалоговое окно **Browse**.

- ⑩ **УЧИТЕСЬ УЧИТЬСЯ!** Меню **Insert** (Вставка) содержит и другие команды для вставки объектов, кроме описанных в этом параграфе. Используя систему поддержки, найдите назначение следующих команд: **Online Pictures** (Изображения из Интернета), **Screenshot** (Снимок экрана), **Online Video** (Видео из Интернета), **Hyperlink** (Гиперссылка).



## 1.7. Форматирование изображений

*Ключевые термины:*

- свойства изображений
- операции над изображениями

Документы, содержащие только текст, не привлекают внимание читателя. Изображения более наглядны и в большинстве случаев могут заменять целые страницы текста. В принципе, изображения можно вставлять в документы, как и любые другие объекты: аудио- и видеофрагменты, формулы и т.п. Поскольку в предназначенных для печати документах изображения играют особую роль, приложение **Word** содержит специальные инструменты для вставки и форматирования изображений.

Следующие команды из меню **Insert** (Вставка) используются для указания источников, из которых берутся изображения для вставки:

**Pictures** (Рисунки) - из файла, указанного пользователем. Этот файл может находиться на локальном компьютере или на одном из компьютеров, подключенных к сети. Очевидно, что для доступа к этому компьютеру пользователь должен обладать этими правами;

**Online Pictures** - Рисунки из Интернета или с виртуальных дисков пользователя;

**Shapes** - из библиотеки заранее подготовленных графических объектов (*shape* - форма, фигура);

**SmartArt** - из приложения **Microsoft Smart Art**;

**Chart** - из приложения **Microsoft Graph Chart** (*chart* - диаграмма, график).

Вставляемые в документ изображения характеризуются следующими свойствами:

- цветом, яркостью и контрастом;
- положением на странице;
- размерами;
- способом обтекания изображения текстом;
- границами и тенью.

**Процесс задания свойств изображений называется форматированием изображений.**

Для форматирования изображение необходимо выделить. В зависимости от типа выделенного объекта появится меню **Picture Tools Format**, содержащее несколько команд для форматирования изображений (рис. 1.19). Подчеркнем, что многие из этих команд также можно найти в контекстном меню, его можно отобразить, щелкнув правой кнопкой мыши по формируемому изображению.

Команды из меню, предназначенных для форматирования изображений, позволяют настроить параметры отображения рисунка: цвет (**Color**), яркость (**Brightness**) и контраст (**Contrast**). При необходимости края изображения можно обрезать (**Crop**).

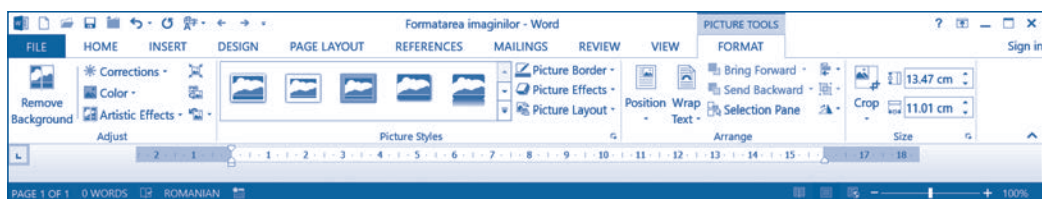


Рис. 1.19. Меню **Picture Tools Format**

Положение изображения устанавливается с помощью страницы **Position** из диалогового окна **Layout** (рис. 1.20).

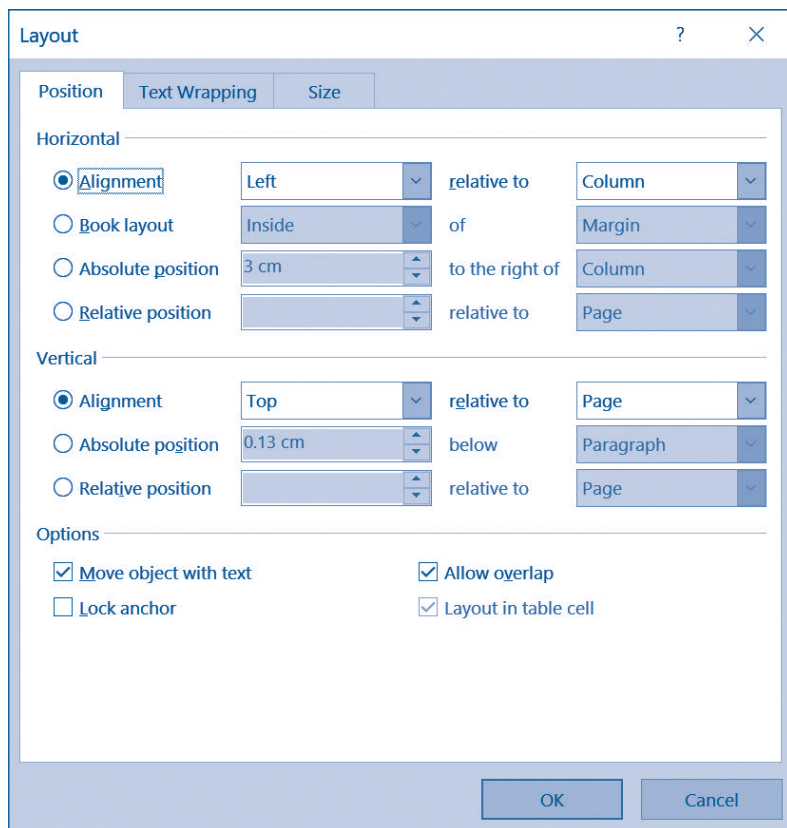


Рис. 1.20. Страница **Position**

Элементы управления рассматриваемой страницы имеют следующее назначение:

**Alignment** (Выравнивание) – выравнивание изображения по горизонтали и вертикали с использованием в качестве ориентиров соответственно боковых и вертикальных краев страницы или текстовых столбцов.

**Absolute position** (Абсолютное положение) – положение изображения определяется в единицах измерения, используемых операционной системой или приложением Word, обычно в сантиметрах.

**Relative position** (Относительное положение) – положение изображения определяется в относительных единицах, обычно в процентах от размера страницы.

**Move object with text** (Перемещать вместе с текстом). Установка этого флажка связывает изображение с определенным абзацем текста, причем соответствующая связь представляется в виде якоря. Перемещение абзаца, в котором закреплено изображение, вызывает автоматическое изменение положения самого изображения. Например, рисунки данного учебника закреплены в абзацах, в которых они впервые упоминаются.

**Lock anchor** (Установить привязку). Когда этот флажок не установлен, пользователь может изменить место привязки, например на другой абзац. Установка флажка закрепляет изображение в соответствующем абзаце, позволяя избежать ошибок позиционирования.

**Способ обтекания текстом изображения** устанавливается с помощью страницы **Text Wrapping** (рис.1.21).

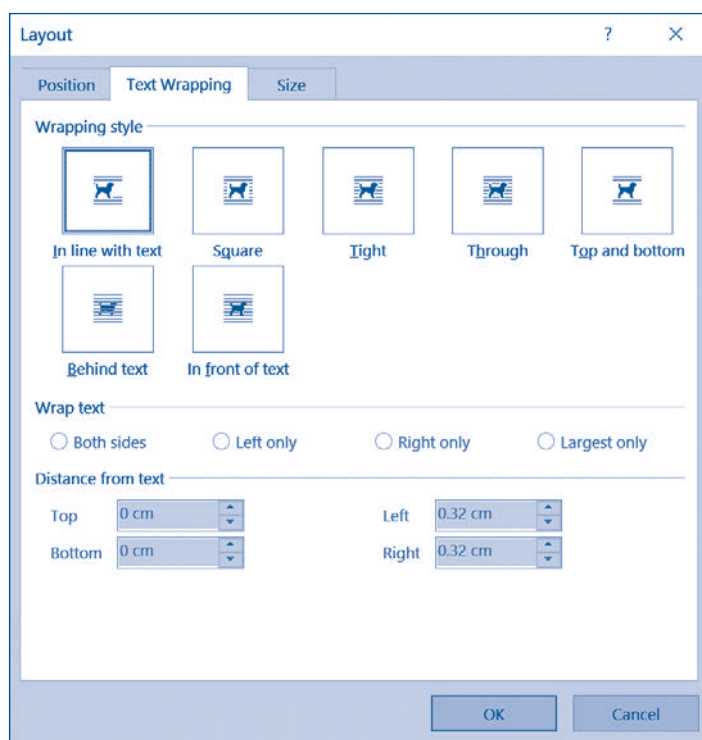


Рис. 1.21. Страница **Text Wrapping**

Раздел **Wrapping style** позволяет выбрать стиль размещаемого текста.

**In line with text** (В строке с текстом). В этом случае изображение ведет себя как простой текстовый символ.

**Square** (Квадрат) – изображение будет внутри воображаемого квадрата, который «обернут» текстом.

**Tight** (Плотно) – текст обтекает изображение, которое не обязательно является прямоугольником, по его контуру. Таким образом, если изображение имеет треугольную форму, оно будет внутри воображаемого треугольника, «обернутого» текстом.

**Through** (Через) – через текст. Изображение прервет «поток текста». В этом случае при чтении взгляд читателя должен будет «прыгать» через изображение.

**Top and bottom** (Сверху и снизу) – текст будет «обертывать» изображение только сверху и снизу, без использования свободных пространств слева и справа.

**Behind text** (За текстом) – изображение будет позади текста, сам текст будет перекрывать его.

**In front of text** (Перед текстом) – изображение будет перед текстом, текст под ним будет невозможно прочесть.

В разделе **Wrap to** (Текст) указываются части изображения, которые будут окружены текстом: вокруг (**Both sides**), слева (**Left only**), справа (**Right only**) или по большей стороне (**Largest only**). Расстояние между текстом и изображением устанавливается четырьмя счетчиками раздела **Distance from text** (Расстояние от текста).

**Размеры изображения** можно изменить с помощью страницы **Size** (Размер), а цвет фона и границ – с помощью страницы **Colors and Lines** (Цвета и линии) соответствующих диалоговых окон.

**Операции**, которые можно совершать над изображениями:

- задание свойств (положения и размеров);
- изменение размеров;
- форматирование границ;
- копирование;
- перемещение;
- удаление;
- закрепление.

Приложение **Word** предлагает несколько возможностей для выполнения этих операций:

- через пункты меню **Insert, Format** и с помощью контекстных меню;
- с помощью кнопок стандартных закладок инструментов;
- с помощью элементов управления окон **Format Picture, Format Shape, Format SmartArt Tools, Format Object**;
- с помощью контекстных меню, появляющихся при выборе объектов щелчком правой кнопки мыши;
- с помощью мыши.

Отметим, что с помощью мыши можно выполнять простым и интуитивно ясным способом большинство операций над изображениями. Например, изменить размеры изображения можно путем перетягивания маркера “↔” в нужном направлении, положение всего изображения – приемом “перетяни-и-отпусти”.

В ходе вставки и редактирования изображений должны соблюдаться следующие правила:

1. Документы не должны содержать чрезмерное количество изображений. Используйте изображения лишь в тех случаях, когда это улучшает понимание текста. Бесполезные изображения только отвлекают внимание читателя.

2. Размеры и положение изображений важны так же, как и их содержание. Использование одного большого изображения часто предпочтительнее множества маленьких изображений.

3. Избегайте беспорядочного размещения изображений на странице. Выравнивайте изображения относительно отступов абзацев или наоборот. Маленькое изображение, повторяющееся на всех страницах документа в одном и том же месте страницы, придает документу единый внешний вид (стиль).

## Вопросы и упражнения

- ❶ Как вы думаете, какова роль изображений в составе документа?
- ❷ Где можно взять изображения, предназначенные для вставки в документ?
- ❸ Перечислите свойства изображений. Как вывести указанные свойства на экран?
- ❹ Объясните термин *форматирование изображений*. Какие операции можно выполнять над изображениями?
- ❺ **ЭКСПЕРИМЕНТИРУЙТЕ!** Как ведут себя изображения без фиксированного положения на странице при изменении текста?
- ❻ Известно, что изображения с фиксированным положением на странице могут быть закрепленными и незакрепленными. Как ведут себя такие изображения в случае больших изменений текста?
- ❼ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Выведите на экран диалоговое окно, в котором можно задать способ расположения текста вокруг изображения. Попробуйте найти в этом учебнике изображения, обтекание которых текстом соответствует схемам, приведенным в соответствующем окне.
- ❽ Перечислите возможности **Microsoft Word** по вставке и форматированию изображений. Как вы считаете, какие из них более наглядные и простые?
- ❾ Вставка и форматирование изображений предполагают соблюдение определенных правил. Каковы эти правила?
- ❿ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Соблюдаются ли правила вставки и форматирования изображений в данном учебнике? Аргументируйте ответ.
- ⓫ **СОЗДАЙТЕ!** Вставьте в ранее созданные документы изображения, которые отражали бы смысл соответствующих произведений. Например, в текст **Глосса** – изображение песочных часов.
- ⓬ **СОЗДАЙТЕ!** Пользуясь инструментами приложения **Word**, создайте следующие документы:



- 13 **ЭКСПЕРИМЕНТИРУЙТЕ!** Используя систему поддержки **Word** и поисковую систему в Интернете, найдите назначение всех команд **Word** для форматирования изображений. Примените каждый из этих инструментов форматирования к изображениям в разрабатываемых документах и наблюдайте за эффектами соответствующих действий.
- 14 **УЧИТЕСЬ УЧИТЬСЯ!** Надписи (*text box*) представляют собой области страницы, размеры и положение которых устанавливаются пользователем. Для объектов внутри надписи (текста, таблиц, изображений) могут быть установлены собственные параметры форматирования. Выведите на экран свойства надписей. Определите, какие операции можно выполнить над ними. Как вы считаете, в каких случаях оправдано применение надписей?

## 1.8. Объектно-ориентированная графика

*Ключевые термины:*

- точно-ориентированная графика
- объектно-ориентированная графика

Известно, что перед началом компьютерной обработки изображения нужно разбить на микрозоны, называемые точками или пикселями. Каждая точка представляется в памяти компьютера одним (для монохромного изображения) или тремя двоичными словами (для цветного изображения), а обработка изображения сводится к изменению соответствующих слов.

**Представление и обработка изображений путем их деления на микрозоны называется точно-ориентированной, или растровой графикой.**

Точно-ориентированная графика используется в приложении **Paint**, которое хранит изображения в файлах с расширением **.bmp** (bit map “карта битов”). Естественно, при изменении размеров изображения увеличиваются или уменьшаются все микрозоны, из которых оно состоит. Главный недостаток точно-ориентированной графики в том, что изменение размеров изображений ухудшает их качество. Для примера на *рис. 1.22, а)* представлены созданные в приложении **Paint** изображения, размеры которых изменены.

Лучшее качество изображений обеспечивается объектно-ориентированной графикой. Сложные изображения в ней формируются из более простых графических объектов: линий, квадратов, прямоугольников, окружностей, эллипсов и т.п.

**Представление и обработка изображений путем их деления на более простые графические объекты называется объектно-ориентированной, или векторной графикой.**

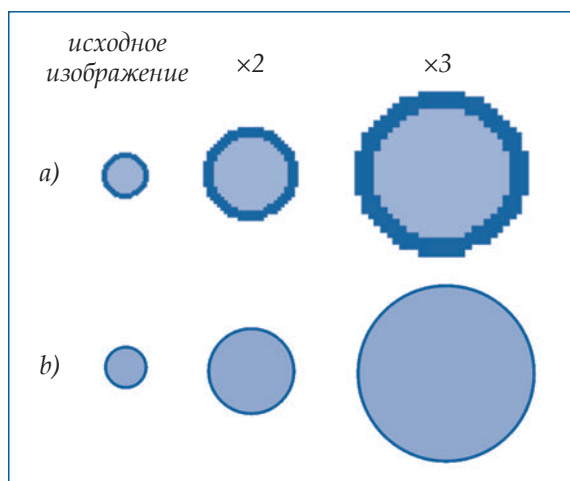


Рис. 1.22. Изменение размеров изображений:  
*a* – точно-ориентированная графика; *b* – объектно-ориентированная графика

Изображение, реализованное методом объектно-ориентированной графики, представляется в компьютере в виде списка. Каждый элемент списка содержит всю необходимую информацию для вычерчивания соответствующего объекта: координаты центра и радиус каждого круга, координаты вершин каждого прямоугольника и т.п. Обработка изображений реализуется путем пересчета координат и размеров каждого графического объекта из списка. Для вывода изображения на экран или на печать компьютер просматривает списки и “рисует” каждый графический объект. Рисование реализуется установкой яркости и цвета каждой микрзоны экрана или принтера. Поскольку размеры микрзон больше не зависят от размеров графических объектов, качество изображений не меняется при изменении их размеров (рис. 1.22, *b*).

Приложение **Word** содержит отдельное меню **Shapes** для работы с объектно-ориентированной графикой. Доступ к возможностям этого меню осуществляется с помощью диалогового окна, которое появляется при выполнении команды **Insert, Shapes** (рис 1.23).

Инструменты из этого окна позволяют рисовать следующие графические объекты:

- линии;
- квадраты и прямоугольники;
- окружности и эллипсы;
- сегменты окружностей или эллипсов;
- произвольные фигуры;
- выноски;
- автофигуры.

Нужный графический объект выбирается щелчком мыши на нужный элемент. Отметим, что каждый графический объект, независимо от типа, занимает на странице фиксированное положение.



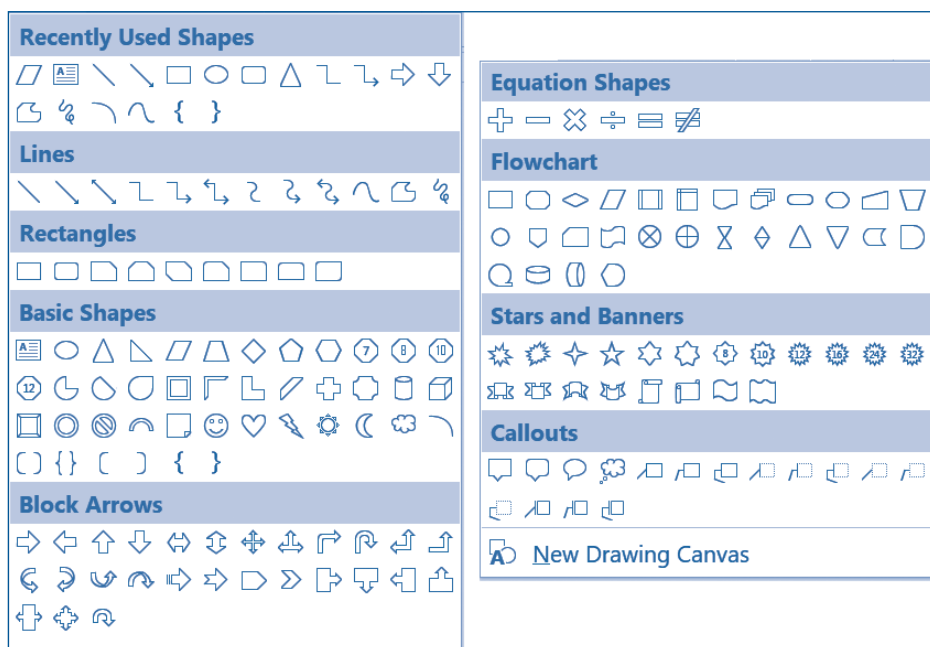


Рис. 1.23. Инструменты рисования в объектно-ориентированной графике

Для создания сложных изображений пользователь последовательно вставляет в рисунки нужные объекты: линии, квадраты, окружности и т.п. Сразу после вставки устанавливаются свойства каждого объекта: толщина и цвет линий границы, цвет заполнения, специальные эффекты. В ходе вставки графические объекты могут накладываться друг на друга. **Порядок** наложения объектов может быть изменен с помощью команд **Bring to Front** и **Sent to Back**, которые позволяют переместить на передний или задний план нужный объект (рис. 1.24).

Другой часто используемой операцией при работе с изображениями является **группировка** объектов. Операция позволяет обрабатывать несколько графических объектов как один составной объект. После выполнения группировки графические объекты, составляющие группу, могут быть:

- увеличены или уменьшены с сохранением взаимных пропорций;
- перемещены с сохранением взаимного положения;
- стерты;
- скопированы из одного документа в другой.

При необходимости пользователь может разгруппировать объекты, сделав их снова независимыми.

Контекстное меню **Format Shape** (рис. 1.25) предлагает и другие возможности для обработки графических объектов:

- штриховка (**Shadow**);
- отражение (**Reflection**);
- свечение (**Glow**);
- трехмерное форматирование (**3-D Format**);



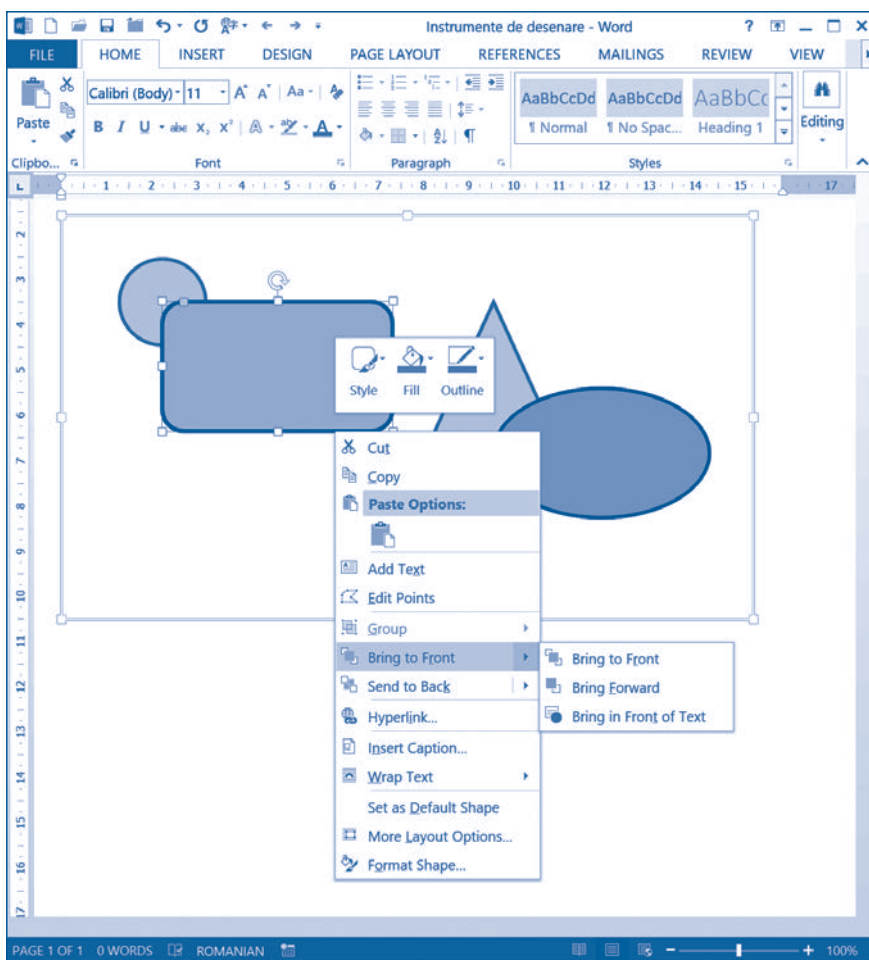


Рис. 1.24. Команда **Bring to Front**

- вращение (**Rotate**);
- отражение относительно вертикали или горизонтали (**Flip**);
- выравнивание (**Align**).

Выравнивание можно осуществлять относительно горизонтальных или вертикальных линий сетки, называемых канвой, относительно краев физической страницы или других графических объектов. (рис. 1.25).

**Выноски (Callouts)** представляют собой особый тип графических объектов, используемых для пояснений к определенным элементам документа (рис. 1.26).

Выноски состоят из надписи и связи с поясняемым элементом. К тексту внутри надписи могут быть применены собственные параметры форматирования. Текст форматируется с помощью команд из меню **Format** или соответствующих кнопок панели форматирования. Пользователь может вставлять в документ выноски различных форм. Форму выносок можно изменять с помощью мыши.

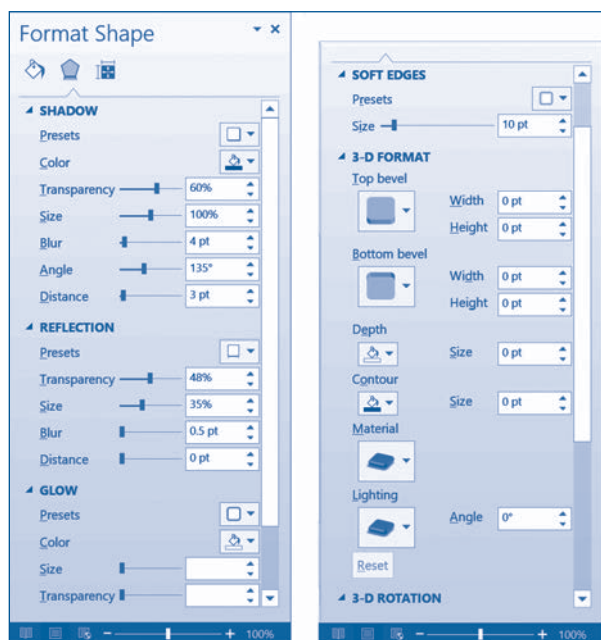


Рис. 1.25. Диалоговое окно **Format Shape**

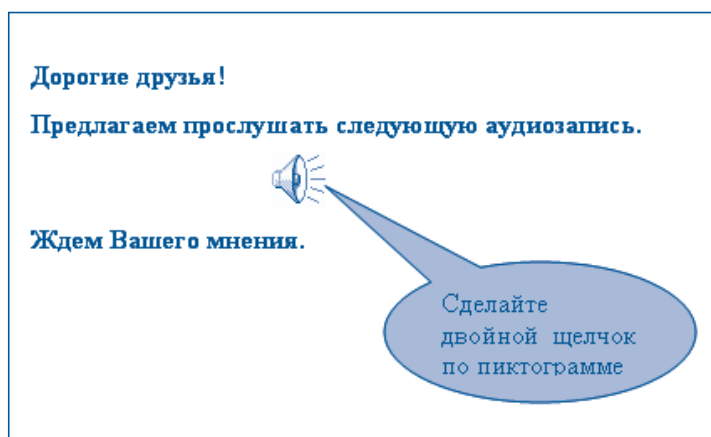


Рис. 1.26. Выноска

## Вопросы и упражнения

- ❶ Объясните термин *точечно-ориентированная графика*. Как кодируются и обрабатываются изображения в этой графике?
- ❷ **ЭКСПЕРИМЕНТИРУЙТЕ!** Нарисуйте в программе **Paint** изображение, представленное на рис. 1.22, а. Увеличьте его в 2, 4 и 6 раз. Как изменяется качество изображения? Объясните причину изменений.

- ③ Объясните термин *объектно-ориентированная графика*. Как кодируются и обрабатываются изображения в такой графике?
- ④ **ЭКСПЕРИМЕНТИРУЙТЕ!** Создайте в программе **Word** изображение, представленное на *рис. 1.22, б*. Увеличьте его в 2, 4 и 6 раз. Меняется ли качество изображения при изменении размеров?
- ⑤ Какие операции осуществляются компьютером при отображении рисунка, выполненного в объектно-ориентированной графике?
- ⑥ **ИССЛЕДУЙТЕ!** Найдите в меню и диалоговых окнах приложения **Word** графические объекты, которые можно использовать для создания изображений.
- ⑦ **УЧИТЕСЬ УЧИТЬСЯ!** Пользуясь справочной системой, найдите назначение всех кнопок в меню и диалоговых окнах с инструментами рисования.
- ⑧ Выведите на экран все автофигуры приложения **Word**. В каких документах можно использовать эти объекты?
- ⑨ Для чего предназначена группировка? В каких случаях применяется эта операция?
- ⑩ Как можно изменить порядок наложения графических объектов? Когда возникает необходимость такого изменения?
- ⑪ **УЧИТЕСЬ УЧИТЬСЯ!** Пользуясь справочной системой, найдите назначение всех команд диалогового окна **Format Shape**. Проверьте, как эти команды воздействуют на графические объекты.
- ⑫ В каких случаях используются выноски? Как их можно форматировать?
- ⑬ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ!** Нарисуйте несколько изображений, которые можно создать как в приложении **Paint** (точечная графика), так и в **Word** (объектно-ориентированная графика). Создайте эти изображения в обоих приложениях. Сравните качество полученных изображений при изменении их размера.
- ⑭ **СОЗДАЙТЕ!** Создайте с помощью приложения **Word** изображения, показанные на *рис. 1.27*.

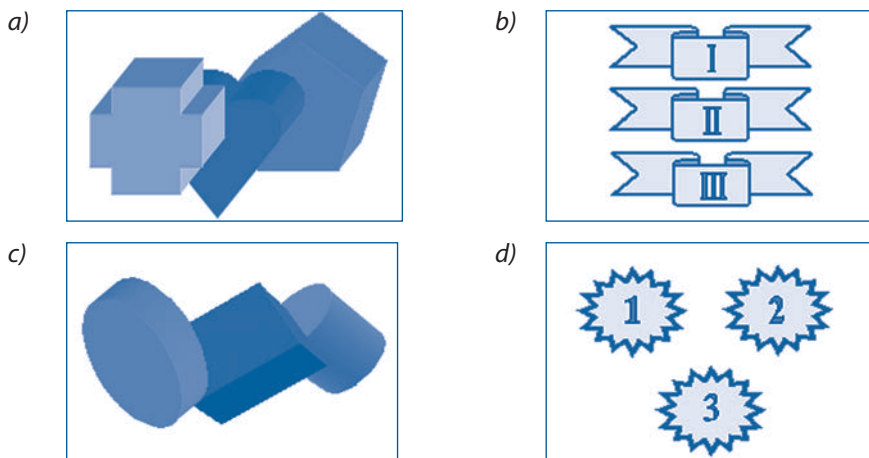


Рис. 1.27. Рисунки, созданные в объектно-ориентированной графике

## 1.9. Диаграммы

*Ключевые термины:*

- диаграмма
- лист данных
- форматирование диаграммы
- тип диаграммы

Тексты, содержащие слишком много чисел, отпугивают большинство читателей. Числовая информация воспринимается намного легче, если она представлена в графической форме.

**Диаграмма – это изображение, в котором числовые значения представлены в виде графических объектов определенных размеров.**

Диаграмма состоит из следующих элементов (рис. 1.28):

- заголовка (не обязателен);
- области рисования;
- оси категорий, как правило, это ось x;
- оси значений, как правило, это ось y;
- индикаторов данных;
- легенды (условных обозначений).

Числовые значения передаются через определенные размеры индикаторов данных: длины, ширины, высоты и т.п.

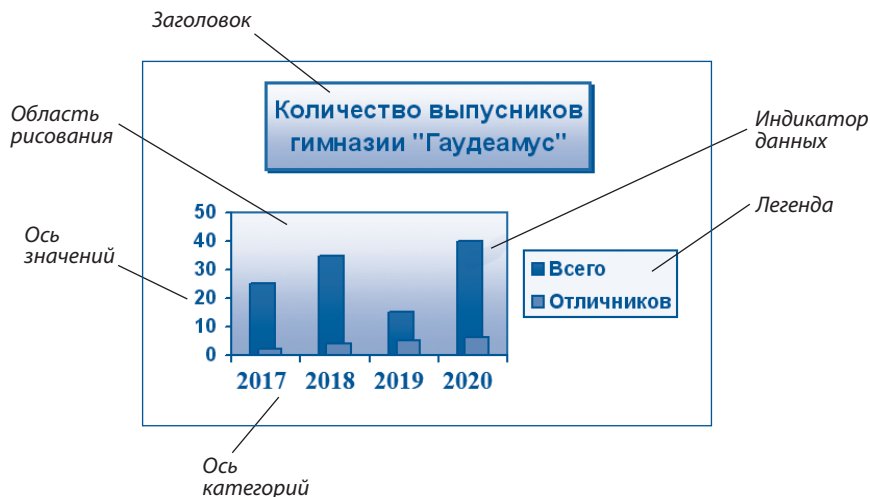
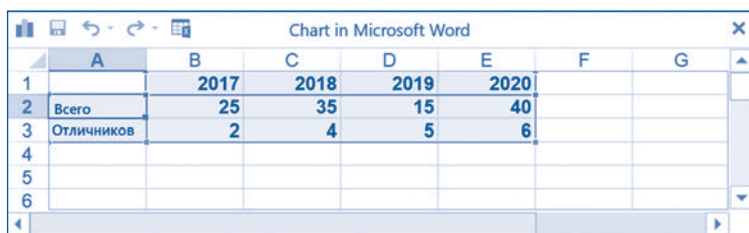


Рис. 1.28. Структура диаграммы

Например, ось категорий на рис. 1.28 содержит категории 2017, 2018, 2019, 2020 или, другими словами, годы выпуска. Ось значений служит для определения количества выпускников и содержит шкалу с числами 0, 10, 20, ..., 50. Индикаторы данных представлены в виде прямоугольников (столбиков), высота которых пропорциональна количеству выпускников соответствующего года.

В диаграмме на *рис.1.28* изображены два ряда данных. Первый ряд соответствует общему количеству выпускников, второй – количеству отличников каждого года выпуска. Для представления данных каждого ряда используются индикаторы, окрашенные в разные цвета. Значения цветов объясняются в легенде.

Диаграммы создаются с помощью приложения **Microsoft Graph Chart**. В этом приложении нужно заполнить особый шаблон, называемый листом данных (*рис. 1.29*).



	A	B	C	D	E	F	G
1		2017	2018	2019	2020		
2	Всего	25	35	15	40		
3	Отличников	2	4	5	6		
4							
5							
6							

*Рис. 1.29. Лист данных*

**Лист данных представляет собой таблицу с данными, которые используются для создания диаграмм.**

Первая строка и столбец листа данных зарезервированы для текста, идентифицирующего информацию в соответствующих ячейках. Остальные ячейки предназначены для самих данных и обозначаются числами и буквами.

Например, первая строка *рис. 1.29* содержит категории 2017, 2018, 2019 и 2020. Первый столбец содержит названия рядов данных – Всего и Отличников. Ячейка, расположенная на пересечении строки 1 и столбца B, содержит количество выпускников 2017 года, равное 25; ячейка из строки 2, столбца C, содержит количество отличников 2018 года выпуска, равное 4, и т. д.

Над листами данных можно выполнять следующие операции:

- изменять, удалять или копировать данные из ячеек;
- удалять или добавлять строки и столбцы;
- изменять размеры столбцов.

Большинство команд, позволяющих выполнять указанные операции, содержат контекстные меню, которые можно вывести на экран, поместив курсор на нужном объекте. Часто используемые операции можно выполнять, пользуясь соответствующими кнопками в диалоговых окнах, появляющихся при выполнении соответствующих команд панели инструментов.

Каждый объект, входящий в состав диаграммы, является относительно независимым и характеризуется определенными свойствами. Например, у **заголовка** есть формат текста, цвет и фон. **Оси** обладают стилем линий, масштабом, форматом чисел, ориентацией и выравниванием текста. **Индикаторы данных** характеризуются формой, стилем, цветом. Одинаковые индикаторы соответствуют данным одного и того же ряда.

**Процесс установки свойств объектов, входящих в состав диаграммы (заголовок, области рисования, осей, легенды, индикаторов данных), называется форматированием диаграммы.**

Перед выполнением любой операции, предназначенной для форматирования диаграмм, необходимо выделить нужный объект. Выбор объекта вызывает появление на ленте меню группы **Chart Tools** (Работа с диаграммами), которая содержит меню **Design** и **Format**. Например, выделение некоторой серии данных и выполнение команды **Format Data Series** приводит к появлению диалогового окна, в котором задаются все свойства выбранного объекта (рис. 1.30).

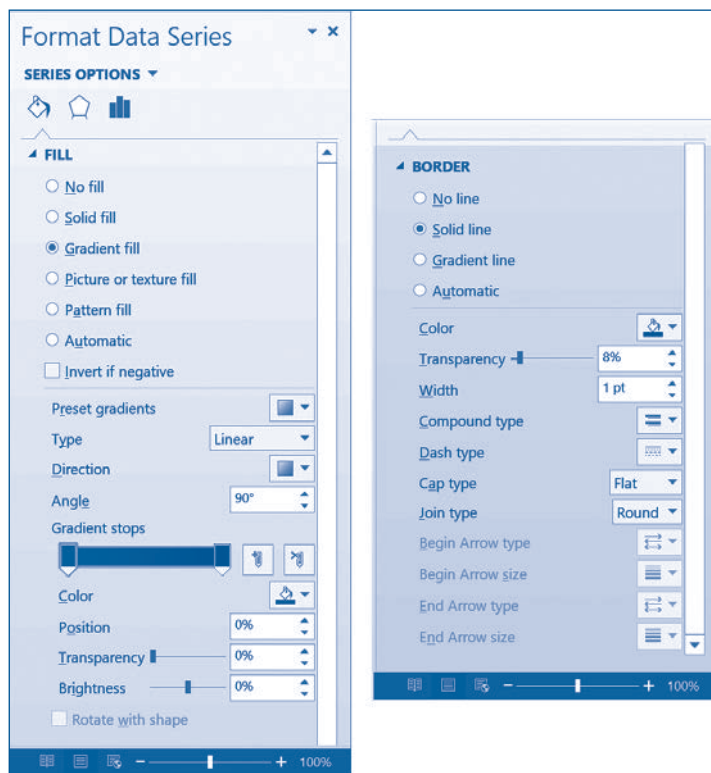


Рис. 1.30. Диалоговое окно **Format Data Series**

Приложение **Microsoft Graph Word** предлагает пользователю разнообразные типы диаграмм. Тип диаграммы определяется графическим объектом, используемым для представления числовых величин: столбцами, линиями, секторами окружности. Наиболее часто используемыми типами диаграмм являются (рис. 1.31):

- гистограммы (столбиковые диаграммы);
- линейчатые диаграммы;
- графики;
- круговые диаграммы.

Естественно, каждый тип диаграмм отображает данные определенным образом. Выбор типа диаграмм осуществляется в соответствии со следующими рекомендациями:

**1. Гистограммы** отображают разные ряды данных, изменяющихся со временем. Например, ежегодное количество выпускников, ежемесячные доходы родителей, объемы продаж коммерческой фирмы и др.

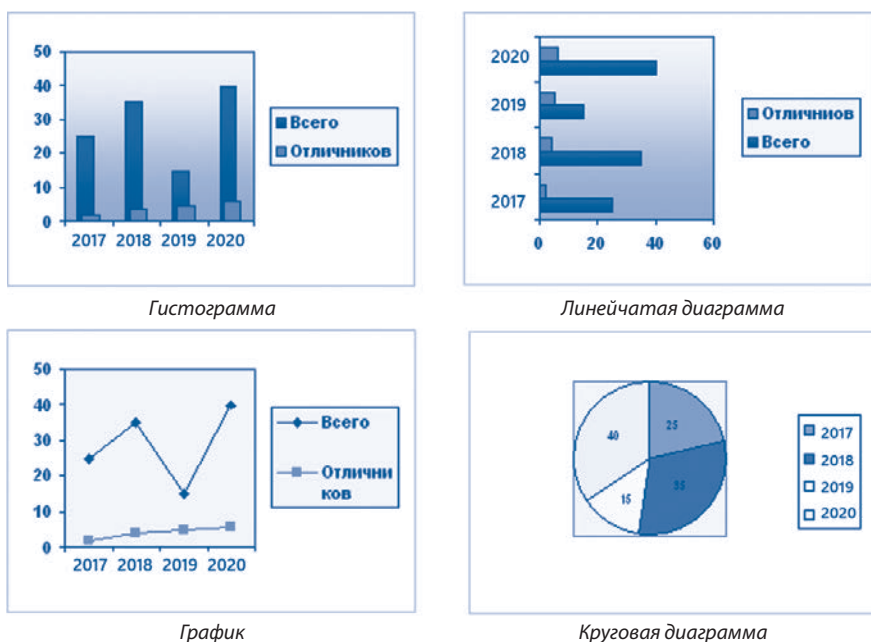


Рис. 1.31. Типы диаграмм

**2. Линейчатые диаграммы** используются, когда необходимо сравнивать данные, не изменяющиеся во времени. Например, для наглядного сопоставления производительности различных моделей компьютеров.

**3. Графики** удобны для представления тенденций или соотношения заданных величин за определенный период времени. Примером таких графиков являются температура пациента в больнице, курс лея по отношению к доллару США.

**4. Круговые диаграммы** используются для отображения взаимоотношений между целым и частями. Например, в кулинарном рецепте каждый сектор представляет количество определенного продукта в блюде.

Для задания типа диаграммы активизируется команда **Change Chart Type** (Изменить тип диаграммы). Отметим, что меню **Design** и **Format** из закладки **Chart Tools** содержат больше возможностей для редактирования диаграмм, которые позволяют настроить положение легенды, отобразить на диаграмме определенные значения, размеры осей, сетку и т.п.

## Вопросы и упражнения

- ИССЛЕДУЙТЕ.** Для чего предназначены диаграммы? Найдите в учебниках истории и географии три диаграммы.
- Перечислите объекты, из которых состоит диаграмма. Какими свойствами они обладают?
- Объясните структуру листа данных. Какие операции можно выполнять над ними?
- Какова связь между листом данных и соответствующей диаграммой?
- Объясните термин *форматирование диаграмм*. Как выполнить такое форматирование?



- ⑥ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ!** Назовите типы диаграмм. В каких случаях используется каждый тип? Приведите примеры.
- ⑦ **ИССЛЕДУЙТЕ!** Определите тип диаграмм, найденных в учебниках истории и географии.
- ⑧ Создайте диаграмму, представленную на *рис. 1.28*. Введите в лист данных количество выпускников вашей гимназии.
- ⑨ **СОЗДАЙТЕ!** Используя приведенную таблицу, создайте гистограмму “Доход фирмы”.

	Январь	Февраль	Март	Апрель	Май	Июнь
Отдел А	70	50	90	20	100	40
Отдел Б	100	150	110	180	30	120

- ⑩ **РАБОТАЙТЕ В КОМАНДЕ!** Создайте линейчатую диаграмму, представляющую средние оценки друзей. Оценки должны отображаться вдоль оси *x*, а имя и фамилия каждого ученика – вдоль оси *y*.
- ⑪ Используя данные из приведенной ниже таблицы, создайте график Курс американского доллара в молдавских леях.

	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Лей	12,30	12,15	11,72	12,06	13,05	15,62	19,66	19,98	17,10	17,14	17,21

- ⑫ **СОЗДАЙТЕ!** Автопарк некоторого предприятия состоит из 60 легковых автомобилей, 20 грузовиков и 15 автобусов. Отобразите состав автопарка круговой диаграммой.
- ⑬ **УЧИТЕСЬ УЧИТЬСЯ!** В приложении **Word** диаграммы могут быть созданы с помощью ранее введенных в документ таблиц. Для этого в соответствующей таблице выделяются нужные данные и копируются в рабочий лист создаваемой диаграммы. Введите в документ таблицы из *упр. 9* и *11*. Создайте на основе этих таблиц гистограммы, диаграммы и круговые диаграммы. Как вы считаете, какие из созданных диаграмм являются более наглядными?

## 1.10. Проверка правописания

Ключевые термины:

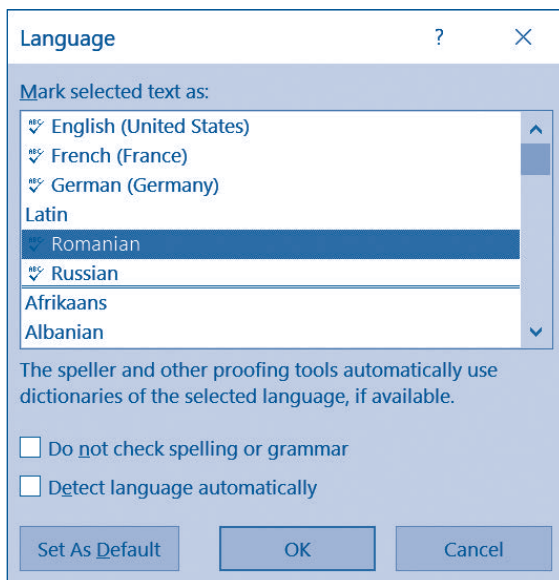
- лексический анализатор
- грамматический анализатор
- тезаурус

В ходе создания и редактирования документов могут появляться ошибки. Приложение **Word** располагает особыми программами, предназначенными для проверки текста. Безусловно, прежде чем выполнить автоматическую проверку произвольного текста, сначала необходимо сообщить программе,



на каком языке он написан. Например, при латинском алфавите компьютер не может однозначно определить языковую принадлежность текста, поскольку указанный алфавит используется во многих языках. Аналогичная ситуация возникает при использовании алфавитов, основанных на кириллице. Поэтому язык всего документа или каждого его отдельного фрагмента должен быть указан явным образом самим пользователем.

Информация о языке, на котором написан фрагмент текста (румынский, английский, русский), указывается с помощью команд **Language**, **Set Proofing Language** (Язык, Выбрать язык) из меню **Review** (Обзор). При этом на экране появляется диалоговое окно, представленное на *рис. 1.32*.



*Рис. 1.32.* Диалоговое окно **Language**

Раскрывающийся список данного окна содержит языки, для которых доступны программы автоматической проверки текста. Очевидно, что соответствующие программы должны быть установлены на компьютере. После указания нужного языка приложение запоминает языковую (лингвистическую) принадлежность выделенного фрагмента текста в параметрах форматирования символов.

**Лексический анализатор – это программа, которая проверяет правописание каждого слова.**

Лексический анализатор сверяет (сравнивает) слова проверяемого текста со словами из списка правильно написанных слов. Такой список называется **словарем**. Подразумевается, что проверяемые слова, которые не найдены в словаре, содержат ошибку.

**Грамматический анализатор – это программа, которая проверяет правописание каждого предложения.**

Работа грамматического анализатора основывается на наборе правил о внутренней структуре слов и сочетании этих слов в предложении.

Для проверки текста обрабатываемого документа активизируется команда **Spelling and Grammar** (Правописание) меню **Review** или нажимается кнопка с тем же названием на панели стандартных инструментов. Программы проверки просматривают текст и в случае ошибки выводят диалоговое окно, в котором указывается ее тип (рис. 1.33).

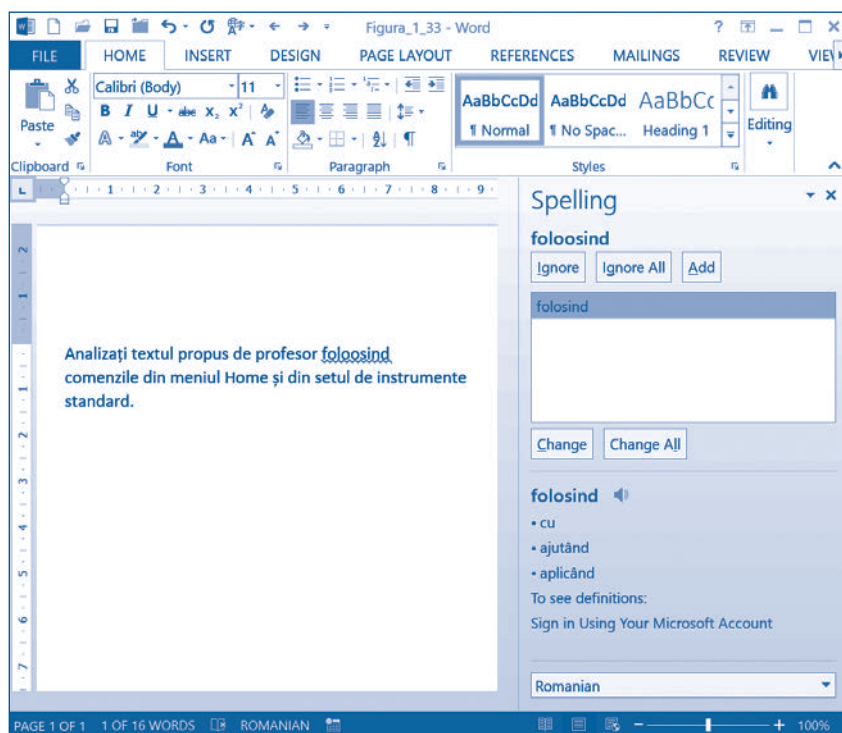


Рис. 1.33. Сообщение о лексических ошибках

Фрагмент текста, который содержит ошибку, выводится в область **Not in Dictionary** (Нет в словаре). Ошибочное слово, а точнее, слово, не найденное в словаре, выделяется волнистой линией красного цвета. Пользователь может исправить его прямо в диалоговом окне или выбрать нужное из списка правильных слов, предлагаемых компьютером в области **Suggestions** (Варианты).

Элементы управления окна **Spelling** имеют следующее назначение:

**Ignore** (Пропустить) – при нажатии этой кнопки происходит переход к проверке текста, следующего за выделенным словом. Используется, когда верно написанное слово не содержится в словаре, например, название населенного пункта или фамилии.

**Ignore All** (Пропустить все) – при следующих обнаружениях выделенного слова сообщения об ошибках появляться не будут.

**Add** (Добавить) – выделенное слово будет добавлено в словарь. В дальней-

шем такое слово будет считаться правильным. Указанная кнопка используется для включения в словарь редко встречаемых слов, например специальных терминов.

**Change** (Изменить) – кнопка нажимается после исправления текста в диалоговом окне. Только тогда изменения из окна будут перенесены в проверяемый текст.

**Change All** (Изменить все) – все следующие вхождения выделенного слова будут исправлены автоматически.

После проверки всех слов текущего предложения грамматический анализатор проверяет предложение целиком. При нарушении правил написания отдельных слов или предложений на экран выводится диалоговое окно, в котором указывается тип ошибки. Например, диалоговое окно на *рис. 1.34* сообщает пользователю о наличии грамматической ошибки **Repeated Word** (Повторяющиеся рядом слова).

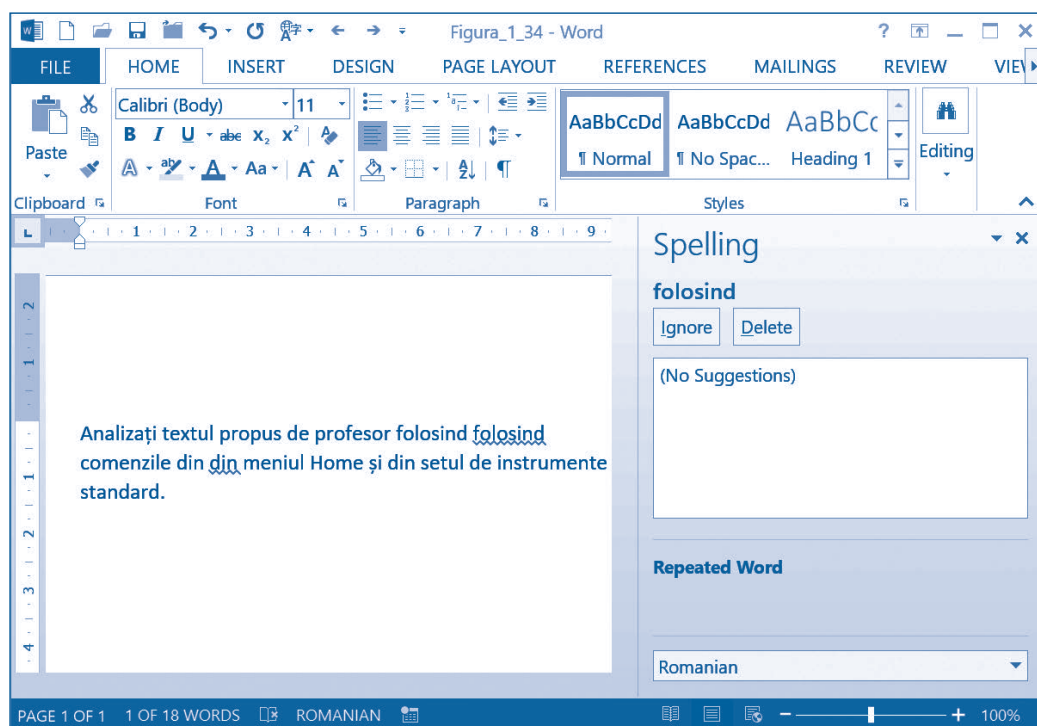


Рис. 1.34. Сообщение о грамматической ошибке

При проверке текста грамматический анализатор подсчитывает следующие **статистические данные** (*рис. 1.35*):

- количество страниц;
- число слов;
- количество знаков без пробелов;
- количество знаков, включая пробелы;
- количество абзацев;
- количество рядов.

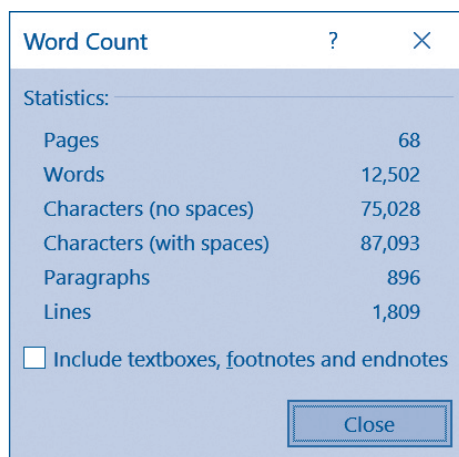


Рис. 1.35. Статистические данные о тексте

Указанные данные отображаются с помощью команды **Word Count** (Счетчик слов) в меню **Spelling and Grammar** (Орфография и грамматика) на ленте меню **Review** (Обзор). Они могут оказаться полезными, когда необходимо адаптировать текст к уровню подготовленности читателя. Например, в коммерческой рекламе должны использоваться короткие предложения, которые быстро читаются и просты по содержанию. Также в случае создания оригинального текста, например художественного произведения, эти данные используются для расчета вознаграждения авторов.

Качество обрабатываемого текста можно улучшить, исключив чрезмерные и раздражающие повторы. Если одно и то же слово встречается в абзаце слишком часто, рекомендуется заменять его синонимами или родственными словами. С этой целью применяется программа **Thesaurus** (Тезаурус), которую можно вызвать, запустив одноименную команду из меню **Review** (Обзор), закладка **Proofing** (Проверка).

**Тезаурус – это программа, которая заменяет выделенные слова синонимами или родственными словами.**

Для замены выделенного слова активизируется команда **Language, Thesaurus** из меню **Review** (Обзор), закладка **Proofing** (Проверка). Диалоговое окно **Thesaurus** (рис. 1.36) содержит выделенное слово (**Looked Up**), его смысл (**Meanings**) и список синонимов (**Replace with Synonym**).

Для замены выделенного слова нажимается кнопка **Insert** (Вставить), а чтобы скопировать в буфер – кнопка **Copy** (Копировать).

Отметим, что правила грамматики, относительно легко воспринимаемые людьми, переводятся на язык, понятный компьютеру, очень тяжело. Компьютер не в состоянии “догадываться” о смысле текста, особенно в тех случаях, когда в нем содержатся ошибки. Следовательно, подсказки, предлагаемые программами проверки текстов, иногда являются неприемлемыми. Поэтому существование компьютерных программ, предназначенных для проверки текстов, не освобождает учащихся от необходимости глубокого изучения родного и иностранных языков.

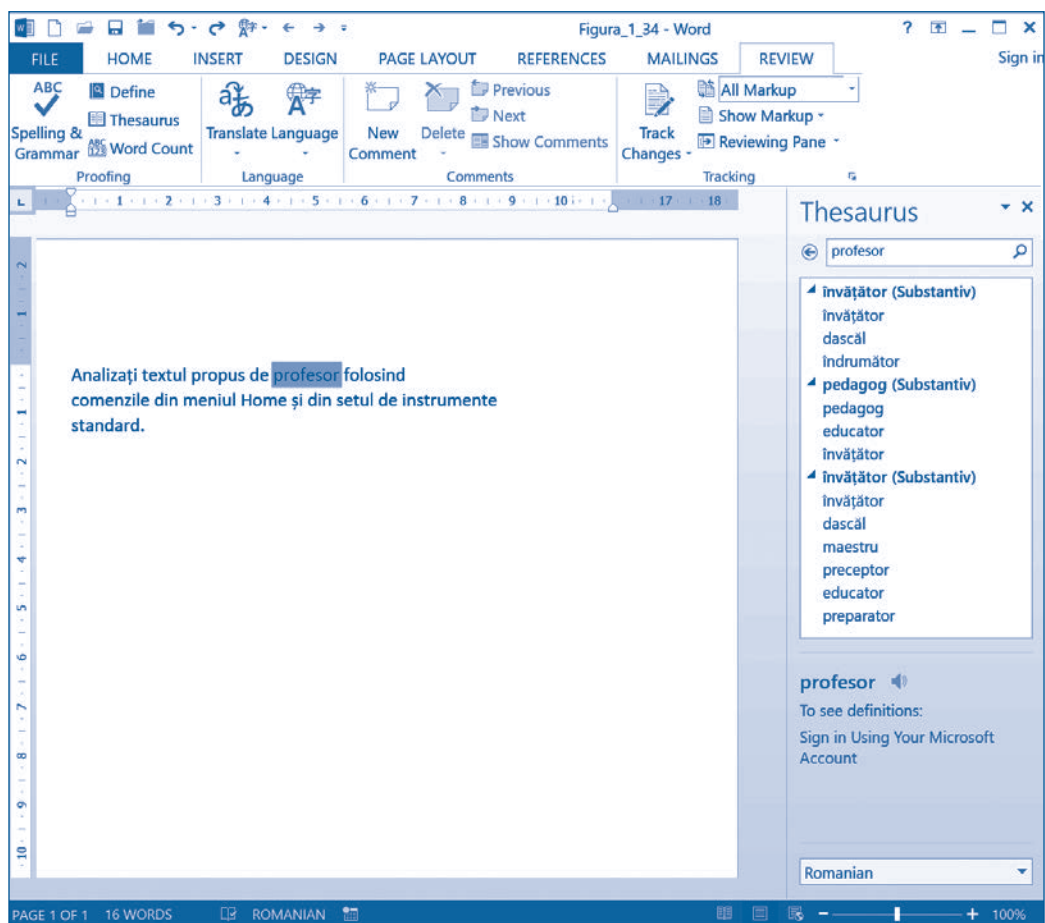


Рис. 1.36. Диалоговое окно **Thesaurus**

## Вопросы и упражнения

- ❶ Как вы считаете, для чего указывается языковая принадлежность каждого фрагмента текста?
- ❷ Объясните термины *лексический анализатор* и *грамматический анализатор*.
- ❸ Как сообщить компьютеру, на каком языке написан определенный фрагмент текста? В каком месте хранится информация об языковой принадлежности текстов?
- ❹ Для чего предназначен словарь, входящий в состав лексического анализатора? Какая информация содержится в рассматриваемом словаре?
- ❺ Какие операции выполняет компьютер при лексической проверке текста?
- ❻ Какая информация необходима для работы грамматического анализатора? Какие операции выполняет рассматриваемая программа в ходе грамматической проверки?
- ❼ **СОЗДАЙТЕ!** Создайте документ, содержащий тексты *упр. 1, 2 и 3*. Выведите на экран языковую принадлежность каждого абзаца. Проверьте текст и сохраните документ в файле **Uprajnenie**.

- 8 **ЭКСПЕРИМЕНТИРУЙТЕ!** Внесите в текст документа **Uprajnenie** следующие ошибки:
- повторение одной из букв произвольного слова;
  - повторение двух слов в произвольном предложении;
  - перестановка местами двух слов произвольного предложения;
  - добавление произвольного слова в одном из предложений.
- Запустите программы проверки текста и объясните выводимые на экран сообщения. Исправьте текст в соответствии со сделанными компьютером подсказками.
- 9 Для чего предназначен **Thesaurus**? Когда используется эта программа?
- 10 Исправьте тексты в файлах, предложенных преподавателем.
- 11 Проверьте и исправьте, при необходимости, тексты ранее созданных документов **Zvezda, Alisa, Otrocestvo**.
- 12 Наберите один из текстов, изучаемых на уроках иностранного языка, и проверьте его с помощью компьютера.
- 13 Установите следующую языковую принадлежность абзацев документа **Uprajnenie**:  
 упр. 1 – русский язык,  
 упр. 2 – румынский язык,  
 упр. 3 – английский язык.  
 Попробуйте проверить текст данного документа с помощью компьютера. Объясните сообщения, выводимые при проверке на экран.
- 14 **УЧИТЕСЬ УЧИТЬСЯ!** Используя систему поддержки, самостоятельно изучите возможности автоматического перевода текста, предлагаемые приложением **Word**. Переведите тексты, которые вы изучали на уроках иностранного языка. Оцените качество переводов, сделанных приложением **Word**.

## 1.11. Вставка формул

*Ключевые термины:*

- формула
- палитра символов
- палитра шаблонов

На обычном языке **формулы** представляют собой комбинации из букв, цифр и математических знаков, которые однозначно задают математические утверждения или указывают порядок выполнения определенных математических операций. В качестве примера приведем несколько формул, изучаемых на уроках физики и математики:

– среднее арифметическое  $n$  чисел:  $\bar{x} = \frac{x_1 + x_2 + \dots + x_n}{n}$ ;

– объединение множеств  $A$  и  $B$ :  $A \cup B = \{x \in A \text{ или } x \in B\}$ ;

– вес тела:  $G = mg$ ;

– коэффициент жесткости пружины:  $k = \frac{F}{\Delta x}$ ;

– плотность тела:  $\rho = \frac{m}{V}$ .

Редактор текста **Word** предоставляет пользователю два метода вставки формул в документы:

- 1) в виде обычных фрагментов текста;
- 2) в виде объектов, созданных с помощью специальных команд.

Первый метод используется для простых формул, когда все появляющиеся в них символы можно найти на клавиатуре или вставить с помощью команды **Insert, Symbol** (рис. 1.37).

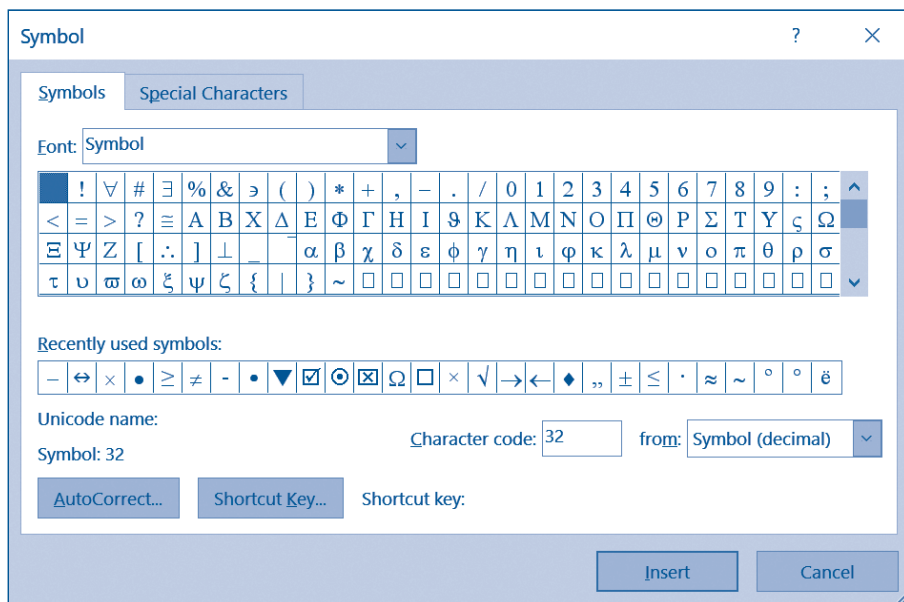


Рис. 1.37. Диалоговое окно **Symbol**

Второй метод предполагает использование команды **Insert, Equation**. Сразу после ее запуска на ленте меню появляется меню **Design** из группы **Equation Tools** (рис. 1.38).

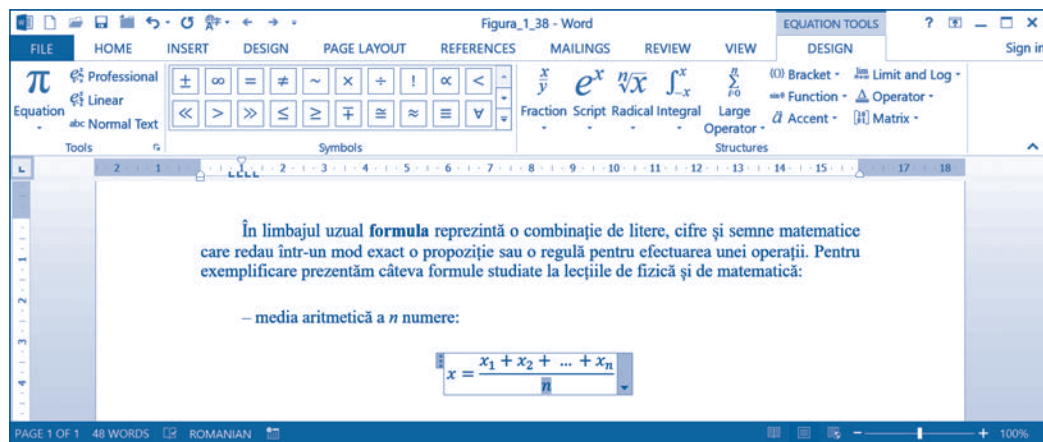


Рис. 1.38. Инструменты меню **Design** из группы **Equation Tools**



Команды из этого меню предоставляют доступ к самым разнообразным коллекциям математических символов и шаблонов, называемых палитрами. **Палитра символов** представляет дополнение к клавиатуре и содержит набор математических знаков, например  $\approx$ ,  $\equiv$ ,  $\pm$ . Выбранный из палитры символ автоматически вставляется в формулу. **Палитра шаблонов** содержит набор полей для записи индексов, дробей, корней и других часто встречающихся в формулах обозначений из математики, физики, химии и т.д. Выбор любого шаблона приводит к автоматической вставке в формулу соответствующих полей, которые позже можно заполнить буквами, цифрами, математическими символами или другими шаблонами.

## Вопросы и упражнения

- 1 Объясните методы вставки формул в документы. Каковы достоинства и недостатки каждого метода?
- 2 **ИССЛЕДУЙТЕ!** Найдите с помощью справочной системы назначение всех команд меню **Equation Tools**. Познакомьтесь с набором символов и шаблонов рассматриваемого приложения.
- 3 Создайте следующий документ.

Из данных таблицы замечаем, что для одной и той же пружины выполняется соотношение:

$$\frac{M_1}{\Delta L_1} = \frac{M_2}{\Delta L_2} = \frac{M_3}{\Delta L_3} = \frac{M_4}{\Delta L_4} = k,$$

где  $k$  представляет коэффициент жесткости пружины.

- 4 **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ!** Создайте документ, содержащий три первых предложения данного параграфа. Используйте для вставки формул оба метода: вставка формул как фрагментов обычного текста и как объектов, созданных другими приложениями. Какой из этих методов проще?

## 1.12. Стили и шаблоны

*Ключевые термины:*

- стиль символа
- стиль абзаца
- шаблон

Создание аккуратных и легко читаемых документов предполагает однотипное форматирование всех фрагментов текста, имеющих одно и то же назначение. В частности, в настоящем учебнике такими фрагментами являются:

- заголовок каждой главы;
- заголовок каждого параграфа;



- основной текст;
- названия таблиц и рисунков;
- заголовков вопросов и упражнений;
- вопросы и упражнения.

В принципе, пользователь может выписать для себя свойства каждого из нужных фрагментов текста (шрифт и размеры символов, выравнивание и отступы абзацев, размеры страниц) на вспомогательный листок и применять их вручную для форматирования каждого из фрагментов соответствующего типа по мере их появления в тексте. Однако такой подход является утомительным, особенно при работе с большими документами.

Приложение **Word** предлагает следующие возможности, облегчающие создание и форматирование документов:

- отображение параметров форматирования абзацев и символов;
- форматирование по образцу;
- форматирование с использованием стилей;
- создание и применение шаблонов.

Параметры форматирования абзацев и символов можно вывести на экран с помощью элементов управления в меню приложения **Word**, которые меняют свой внешний вид в зависимости от выбранного объекта. Для различных объектов документа на экран выводятся свойства соответствующих символов и абзацев.

Форматы определенного объекта документа можно **запомнить** и **применить** к другим объектам с помощью кнопки **Format Painter** (Кисть для форматирования, Формат по образцу) ленты меню **Home**. При нажатии кнопки приложение запоминает параметры форматирования текста, внутри которого расположена точка вставки, а указатель принимает форму кисти. Этой кистью необходимо “покрасить” текст, для которого требуется задать формат исходного текста.

**Форматирование с использованием стилей** очень удобно при обработке больших документов. В общем случае различают стили символов и стили абзацев.

**Стилем символа** называется совокупность свойств символов (шрифт, стиль отображения, размер).

**Стилем абзаца** называется совокупность свойств абзаца (выравнивание, отступы, интервалы, спецэффекты, описание символов, из которых он состоит). Каждому стилю присваивается индивидуальное имя.

Название стиля, примененного к абзацу, можно увидеть в режиме **Draft** отображения документа из меню **View** (Вид) в левой части экрана (рис. 1.39).

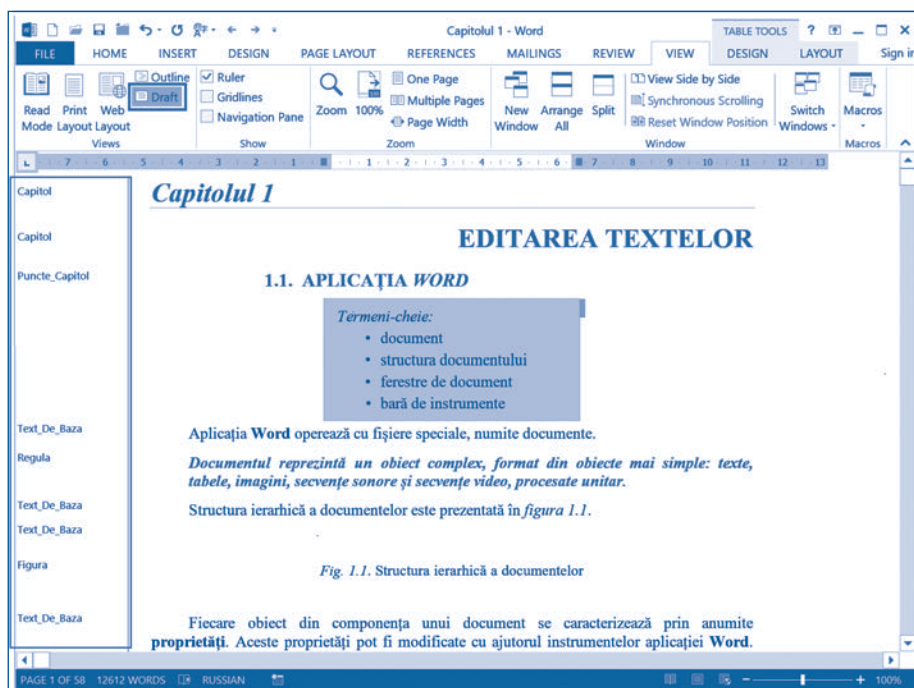


Рис. 1.39. Вывод на экран стилей абзацев текста

Из рисунка 1.39 видно, что при редактировании этого учебника использовались следующие стили:

**Capitol** – стиль абзаца, используемый для форматирования заголовков глав.

**Puncte\_Capitol** – стиль абзаца, используемый для форматирования заголовков разделов в каждой главе.

**Text\_De\_Baza** – стиль абзаца, в соответствии с которым форматируется текст учебника.

**Regula** – стиль абзаца, используемый для выделения правил и определений в учебнике.

**Figura** – стиль, используемый для форматирования названий рисунков.

Подчеркнем, что стиль абзаца содержит свойства как абзаца, так и символов, из которых состоит. Как правило, стили разрабатываются художниками и специалистами по техноредактированию. При необходимости опытные пользователи могут создавать собственные стили. Доступные для использования стили отображаются в раскрывающемся списке **Styles** (Стили) из соответствующей закладки меню **Home** (рис. 1.40).

Для форматирования нужного фрагмента текста пользователь выделяет соответствующую часть документа и выбирает из списка нужный стиль. Название каждого элемента списка стилей отображается на экране тем стилем, который он представляет. Символ “¶” указывает стили абзацев, а символ “a” – стили символов. Кнопки с горизонтальными линиями в закладке **Paragraph** обозначают выравнивание абзаца: по левому, правому краю, по центру или по ширине.

В примере на *рис. 1.40* документ содержит созданные пользователем стили **Adresare**, **Câmp**, **Către**, **Data & Semnătura** и стили **Heading 1** и **Normal**, встроенные в приложение **Word**. Обратите внимание, что **Adresare** – это стиль абзаца, а **Câmp** – стиль символа.

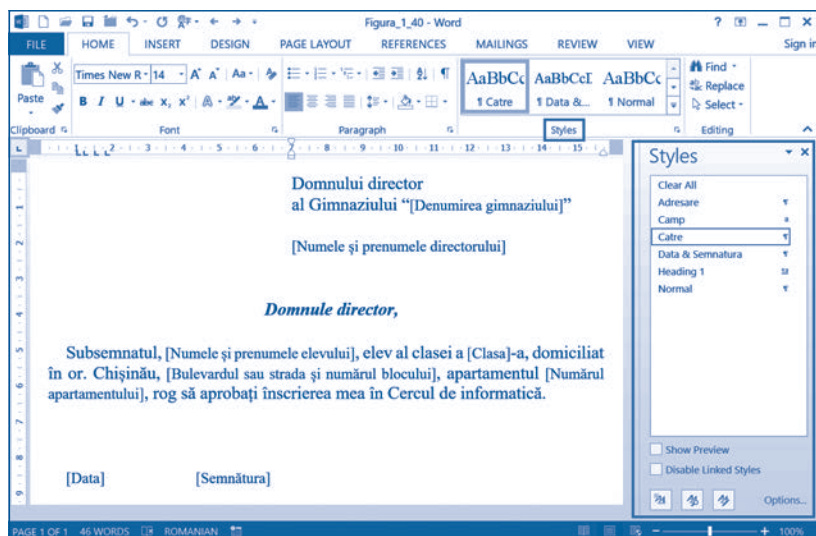


Рис. 1.40. Закладка **Styles** меню **Home**

Свойства стилей в текущем документе можно просмотреть, поместив курсор на их имена в диалоговом окне **Styles**. Изменение свойств каждого из стилей осуществляется с помощью команд контекстного меню, которое отображается на экране при щелчке правой кнопкой мыши по названию нужного стиля (*рис. 1.41*).

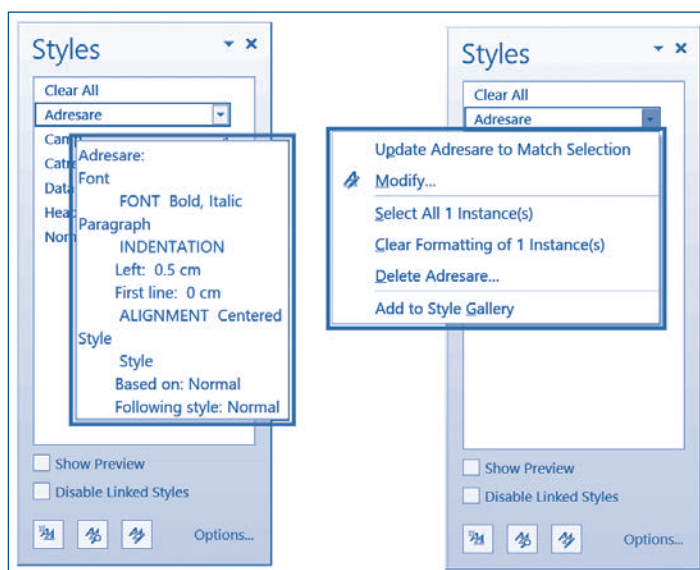


Рис. 1.41. Свойства стиля **Adresare** из связанного с ним контекстного меню

Команды контекстного меню также позволяют удалять существующие стили, создавать новые стили и настраивать их отображение: в алфавитном порядке, все или только некоторые стили в составляемом документе. Более того, в зависимости от конфигурации операционной системы, приложение **Word**, отслеживая форматирование, сделанное пользователем, может автоматически создавать новые стили. Это часто приводит к перегрузке диалогового окна **Styles**. Поэтому в образовательных целях пользователю рекомендуется отключить функцию автоматического создания новых стилей, при необходимости изменив настройки в меню **File, Options**.

Если пользователь не указывает желаемый стиль, приложение будет использовать обычный стиль. Свойства символа и абзаца, определенные в этом стиле, зависят как от конфигурации операционной системы, так и от конфигурации приложения **Word**.

При создании нового документа приложение **Word** должно знать параметры форматирования страниц, абзацев и символов. В принципе, пользователь может повторно вводить указанную информацию для каждого вновь создаваемого документа. Однако практика показывает, что большинство документов имеет одно и то же форматирование и содержит одинаковые фрагменты текста. В качестве примера можно привести заявления, резюме, школьные отчеты, счета на электроэнергию. В таких случаях удобно использовать шаблоны.

Шаблоны представляют собой образцы документов, содержащие форматы страниц, стили абзацев, символов и произвольные объекты (тексты, таблицы, изображения).

Каждый шаблон имеет индивидуальное имя и хранится в файле с расширением **.dotx**. Как правило, файлы наиболее часто используемых шаблонов хранятся в общей папке **Custom Office Templates** (Персонализированные шаблоны сюиты офисных приложений).

Напоминаем, что новый документ создается автоматически, основываясь на шаблоне **Normal**, при запуске приложения **Word**, либо, по желанию пользователя, при выполнении команды **File, New**. Этот шаблон содержит часто используемое форматирование страницы и стиль абзаца с тем же именем. При этом приложение выводит на экран диалоговое окно **New**. При выборе опции **Personal** на экране появятся пиктограммы всех шаблонов, хранящихся в папке **Custom Office Templates** (рис. 1.42).

В этом окне пользователь может выбрать нужный шаблон и создать на его основе новый документ и/или другой шаблон. При обработке файлов приложение **Word** не делает никаких различий между шаблонами и обычными документами. Только при сохранении данных пользователь должен решить, будет ли записываемый файл использоваться в дальнейшем только как документ (расширение **.docx**) или как шаблон (расширение **.dotx**). Для создания шаблонов пользователь устанавливает в диалоговом окне **Save As** параметр **Word Template**.

При выборе опции **Featured** (Рекомендуемые) шаблоны, рекомендованные дизайнерами, будут отображаться в диалоговом окне **New** (рис. 1.43). Их список очень велик, пользователь может создавать документы с различными аспектами. Кроме того, поле поиска диалогового окна **New** позволяет искать нужные шаблоны в Интернете, их количество составляет несколько тысяч.

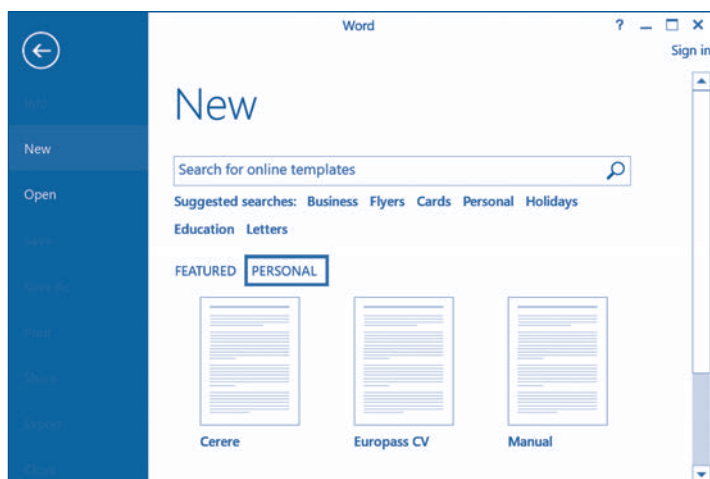


Рис. 1.42. Диалоговое окно **New**

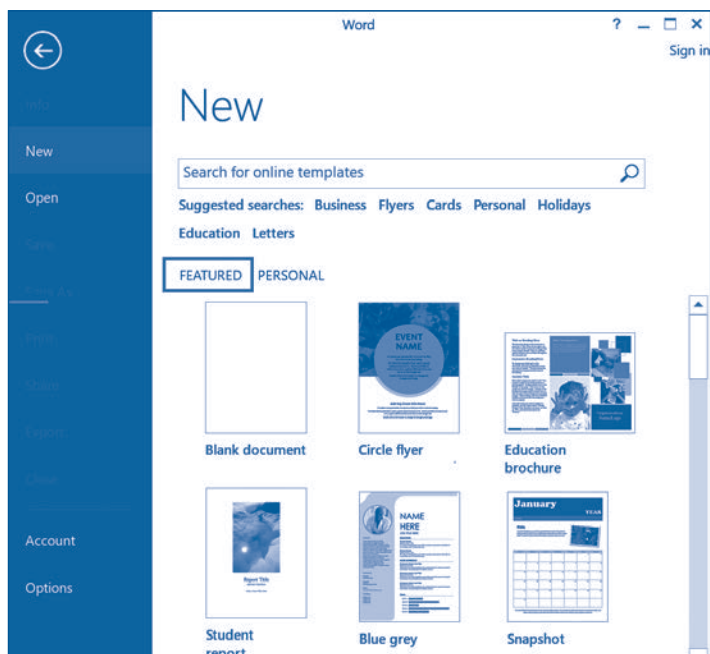


Рис. 1.43. Рекомендованные дизайнерами шаблоны

Стандартные или созданные пользователем шаблоны могут быть использованы также в качестве **библиотеки стилей**. Для этого необходимо нажать кнопку **Manage Styles** (Управление стилями) в диалоговом окне **Styles** и запустить команду **Import/Export**. В соответствующем диалоговом окне указаны файлы, из которых будут экспортированы и импортированы желаемые стили.

Таким образом, пользователь может редактировать текущий документ, используя стили из нескольких шаблонов.

## Вопросы и упражнения

- ❶ Перечислите возможности приложения **Word** по созданию и форматированию документов.
- ❷ **ЭКСПЕРИМЕНТИРУЙТЕ!** Выведите на экран параметры форматирования абзацев и символов в одном из созданных документов.
- ❸ Объясните, для чего предназначена кисть форматирования (форматирование по образцу).
- ❹ **ИССЛЕДУЙТЕ!** Выведите на экран названия стилей, использованных в одном из созданных документов.
- ❺ Объясните термины *стиль абзаца* и *стиль символа*.
- ❻ Для чего предназначены шаблоны? Какая информация содержится в шаблоне?
- ❼ Выведите на экран список стилей шаблона **Normal**.
- ❽ **ИССЛЕДУЙТЕ!** В каждом из данных преподавателем шаблонов определите:
  - список доступных стилей;
  - свойства стилей абзацев;
  - свойства стилей символов.
- ❾ **СОЗДАЙТЕ!** Создайте на основе шаблона **Normal** произвольный документ. Добавьте в список стилей этого документа стили из шаблонов, указанных преподавателем.
- ❿ **УЧИТЕСЬ УЧИТЬСЯ!** При использовании опции **Education** (Образование) в диалоговом окне **New** на экране отображаются шаблоны учебных материалов. Узнайте назначение каждого шаблона. Измените соответствующие шаблоны, адаптируя их к специфике нашей страны и местности, где находится школа, в которой вы учитесь. Используйте модифицированные шаблоны для создания и распространения документов, необходимых вам в школьной жизни: отчетов, писем, объявлений, флаеров и т. д.
- ⓫ **СОЗДАЙТЕ!** Самостоятельно разработайте шаблон, образец которого приведен ниже. Выведите на экран стили абзацев и стили символов, использованных в этом шаблоне.

Директору гимназии  
“[Наименование гимназии]”  
[Фамилия, имя, отчество директора]

От ученика [Класс]-го класса  
[Фамилия, имя, отчество ученика],  
проживающего по адресу  
г. Кишинев, улица [Наименование улицы],  
дом [Номер дома], квартира [Номер квартиры]

Господин директор,

прошу зачислить меня в кружок информатики. С правилами работы в компьютерном классе ознакомлен и обязуюсь их выполнять.

[Дата]                      [Подпись]

- ⓫ **ИССЛЕДУЙТЕ!** Отобразите документ, показанный на *рисунке 1.40*, в режиме просмотра **Draft** (Черновики). Помещая курсор на различные объекты в тексте (символы, слова, строки, абзацы), найдите назначение элементов управления в диалоговом окне **Styles** (Стили).




## 1.13. Комбинированная корреспонденция

Ключевые слова:

- основной документ
- источник данных
- комбинированная корреспонденция

Практика показала, что часто многие документы имеют практически идентичное содержание, разница между ними сводится к двум-трем предложениям. Например, предположим, что создан комитет для организации районной олимпиады по информатике. Комиссией разработана форма (шаблон) приглашения:

	«Учреждение»
	«Адрес»
Дорогие друзья, приглашаем Вас принять участие в районном конкурсе по информатике. Организаторы могут обеспечить проживание и питание «Состав команды» участников.	
Организационный комитет	

Это письмо необходимо отправить примерно в 20 школ района. Данные о каждом школьном учреждении занесены в таблицу вида:

Учреждение	Адрес	Состав команды
Районный теоретический лицей им. М. Губогло	Чадыр-Лунга, ул. Сыртмача, 44	24
Теоретический лицей им. Б. Янакогло	Чадыр-Лунга, с. Копчак, ул. Родака, 24	8
Гимназия им. П. Казмалы	Чадыр-Лунга, ул. Чкалова, 53	3
Теоретический лицей	Чадыр-Лунга, с. Кириет-Лунга, ул. Пушкина, 1	6

Простое, на первый взгляд, решение этой проблемы – написать письмо с указанием названия учреждения, адреса и количества участников теоретического лицея им. М. Губогло. После печати первого письма его можно изменить, указав информацию о теоретическом лицее им. Б. Янакогло. После печати второго письма ввести данные о гимназии им. П. Казмалы и т. д. Очевидно, что отдельное редактирование около 20 писем, по одному для каждой школы, – утомительный и неэффективный процесс.

Гораздо лучшее решение – автоматически создать необходимый набор писем. Для этого используются команды группы меню **Mailings** (Рассылки). Эти команды используют следующие объекты:

**Main Document** (Основной документ) – представляет собой форму (модель) письма. В этой форме информация, которая меняется от одного письма к другому, кодируется с помощью специальных символов, называемых полями. Например, рассмотренная выше форма содержит поля «Учреждение», «Адрес» и «Состав команды».

**Data Source** (Источник данных) – это таблица, в которой первая строка содержит имена полей. Остальные строки таблицы содержат данные, которые будут вставлены вместо полей в основной документ.

**Letters** (комбинированная корреспонденция) – файл Word, содержащий по одному письму для каждой строки таблицы в источнике данных.

Основной документ можно ввести в компьютер, отредактировать и сохранить точно так же, как и другие документы **Word**. Однако, перед тем как вставить в него нужные поля, основной документ необходимо подключить к источнику данных.

Обычно в качестве источника данных используются таблицы, созданные в приложении **Access**. Например, на *рисунке 1.44* представлен источник данных, содержащий список учреждений, ученики которых будут приглашены на районную олимпиаду по информатике.

Учреждение	Адрес	Состав команды
Районный ТЛ им. М. Губогло	Чадыр-Лунга, ул. Сыртмача,	24
ТЛ им. Б. Янаоголо	Чадыр-Лунга, с. Копчак, ул. I	8
Гимназия им. П. Казмалы	Чадыр-Лунга, ул. Чкалова, 5	3
Теоретический лицей	с. Кириет-Лунга, ул. Пушкин	6
*		0

Рис. 1.44. Источник данных

Поля вставляются с помощью команды **Insert Merge Field** (Вставить поле слияния). Поместив курсор в то место, куда вы хотите вставить поле, и запустив соответствующую команду, в появившемся на экране диалоговом окне (*рис. 1.45*) выберите нужное поле.

В целом группа меню **Mailings** (Рассылки) позволяет объединить все операции для создания комбинированной корреспонденции. Основные возможности этого меню:

- создание и редактирование основных документов для писем, конвертов, почтовых марок и электронных писем;
- создание и редактирование источников данных с помощью специальных форм;



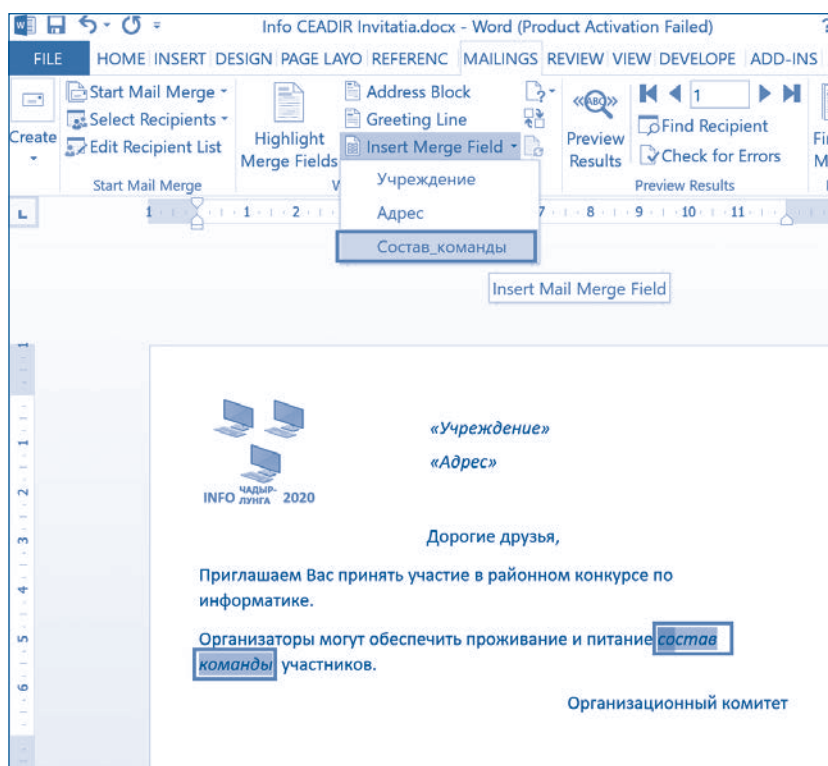


Рис. 1.45. Вставка полей слияния

- использование источников данных, созданных с помощью других приложений операционных систем;
- создание комбинированной корреспонденции только для определенных строк таблицы в источнике данных.

Очевидно, что эти инструменты можно использовать не только для писем, но и для других документов, которые имеют практически идентичное содержание: школьная ситуация каждого ученика, платежные документы, путевые листы и т. д.

## Вопросы и упражнения

- ❶ Приведите примеры документов, которые можно подготовить с помощью средств комбинированной корреспонденции.
- ❷ Объясните термин «основной документ». Чем основной документ отличается от других документов **Word**?
- ❸ Какую информацию содержит источник данных? Как организована эта информация?
- ❹ Объясните термин «поле». Каково назначение полей в документе?
- ❺ Какую информацию содержит документ комбинированной корреспонденции? Какие операции выполняет компьютер для создания такого документа?

- ⑥ **ИССЛЕДУЙТЕ!** Отобразите на экране группу меню **Mailings** (Рассылки). С помощью справочной системы узнайте назначение каждого элемента управления в этих меню.
- ⑦ **СОЗДАЙТЕ!** Изобразите на рисунке потоки данных между объектами комбинированной корреспонденции: основным документом, источником данных и комбинированной корреспонденцией.
- ⑧ **УЧИТЕСЬ УЧИТЬСЯ!** Используя справочную систему, узнайте, как создавать документы для печати конвертов и почтовых марок для автоматической передачи сообщений электронной почты.
- ⑨ **ЭКСПЕРИМЕНТИРУЙТЕ!** Создайте, опираясь на материалы, изученные в этом параграфе, основной документ **Письмо-приглашение** и источник данных **Школьные учреждения**. Сохраните в файле и отобразите на экране объединенную корреспонденцию подобно той, что показана на *рис. 1.46*.

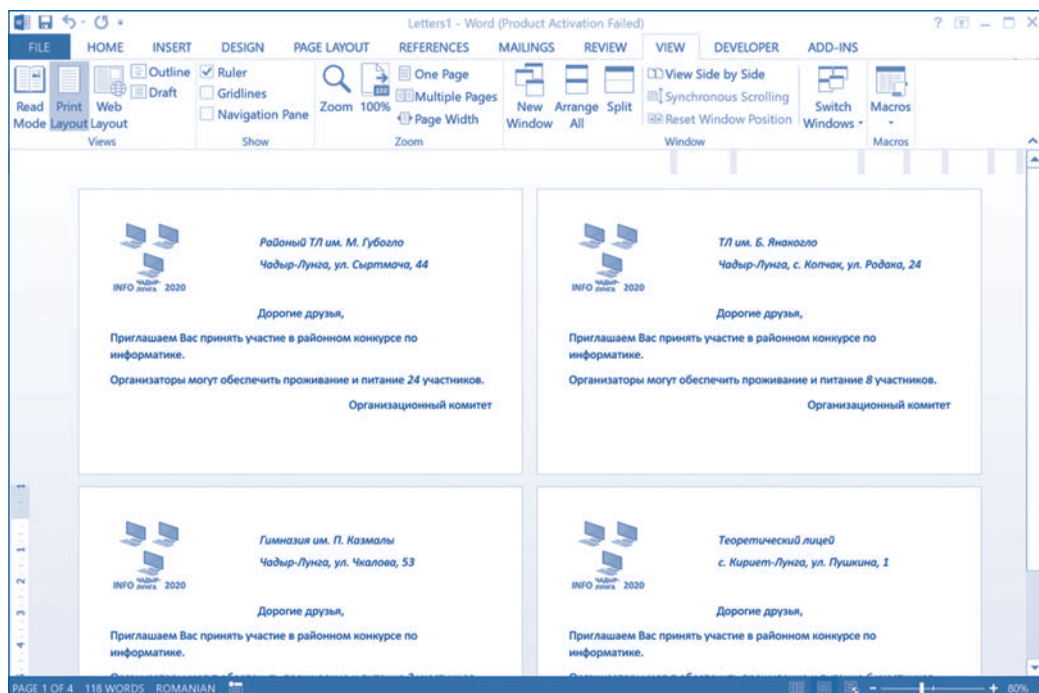


Рис. 1.46. Комбинированная корреспонденция

- ⑩ **РАБОТАЙТЕ В КОМАНДЕ!** Создайте в соответствии со следующей моделью документ **Школьная ситуация** для каждого ученика в классе:

Школьная ситуация	
Ученика (ученицы).....	
1. Русский язык .....	
2. Иностранный язык .....	
3. Математика .....	
4. Физика .....	
5. История румын .....	
Классный руководитель .....	

### 2.1. Алгоритмы и исполнители

*Ключевые термины:* • алгоритм • исполнитель • ручное управление  
• программное управление • программа • язык программирования

Обработка информации с помощью компьютера предполагает наличие определенной программы, которая загружается в его внутреннюю память. Мы уже знаем, что любая компьютерная программа представляет собой последовательность двоичных слов, которые указывают компьютеру порядок вычислений. Компьютерные программы разрабатываются на основе алгоритмов.

**Интуитивно алгоритм можно представить как конечную совокупность правил и предписаний, выполнение которых обеспечивает решение поставленной задачи. Процесс разработки алгоритмов называется алгоритмизацией.**

Напомним, что в повседневной жизни предписания представляют собой точные указания по выполнению определенных действий (определенной работы). Примеры таких указаний:

- 1) до запуска компьютера проверьте подключен ли сетевой кабель;
- 2) если  $a \neq 0$ , вычислите  $x = b/a$ ;
- 3) чтобы найти корни уравнения второй степени, в первую очередь вычислите дискриминант  $D = b^2 - 4ac$ .

Слово алгоритм происходит от имени великого математика средневековья Аль-Хорезми (*Al-Khwarizmi Muhammed ibn Musa*, примерно 780–850 гг.), который впервые сформулировал точные и ясные правила для выполнения основных арифметических операций, которые сейчас изучаются в начальной школе: сложение, вычитание, умножение и деление. Позже эти правила появились под наименованием **алгоритм** в книге «Элементы Евклида». Считается, что одним из первых известных в математике алгоритмов является алгоритм Евклида для нахождения наибольшего общего делителя двух натуральных чисел.

Пример алгоритма:

*Исходные данные:* длины  $a$ ,  $b$  катетов прямоугольного треугольника.

1. Вычислите  $x=a^2$ .
2. Вычислите  $y=b^2$ .

3. Вычислите  $z = x + y$ .

4. Вычислите  $c = \sqrt{z}$ .

*Результат:* длина гипотенузы  $c$ .

Очевидно, что приведенный выше алгоритм может быть выполнен любым человеком, знающим основные арифметические действия (операции), а также операции со степенями и радикалами.

В настоящее время смысл слова алгоритм стал значительно шире. Этот термин используется в различных областях науки и техники для точного и однозначного описания последовательности действий, предназначенных для решения определенной задачи. В информатике понятие алгоритма неразрывно связано с понятием исполнителя.

**Исполнитель представляет собой объект, который может выполнять определенные команды. Множество таких команд образуют систему команд исполнителя.**

Например, телевизор может рассматриваться как объект, который выполняет следующие команды: включение, выключение, увеличение громкости, уменьшение громкости, прием канала №1, прием канала №2 и т.п. Соответствующие команды передаются телевизору с помощью кнопок пульта управления. Аналогичным образом компьютер представляет собой объект, который выполняет следующие команды: сложение, вычитание, умножение, деление и сравнение чисел, чтение данных с клавиатуры, вывод данных на экран, запись данных на магнитный диск, чтение данных с оптического диска и т.п.

Точное определение исполнителя включает:

1) описание системы (набора) команд, которые умеет выполнять («знает») исполнитель;

2) описание среды, в которой работает исполнитель.

В дальнейшем мы будем изучать двух исполнителей – **Кенгуренка** и **Муравья**, разработанных в учебных целях группой учеников и преподавателей нашей страны.

### ***Исполнитель Кенгуренок***

Этот исполнитель представляет собой компьютерную программу, предназначенную для выполнения в операционной системе **Windows**. Окно приложения **Кенгуренок** представлено на рисунке 2.1.

Сам исполнитель представлен пиктограммой маленького кенгуру, который может выполнять следующие команды:

**ШАГ** – кенгуру перемещается на одну клетку, рисуя при этом соответствующий отрезок прямой;

**ПРЫЖОК** – кенгуру перемещается на одну клетку, но ничего не рисует;

**ПОВОРОТ** – кенгуру поворачивается на  $90^\circ$  по часовой стрелке.

Окно приложения **Кенгуренок** состоит из следующих элементов:

1) **Строка меню**, включающая в себя стандартные пункты меню Файл, Правка, Команды, Опции, Помощь.

2) **Центр управления**, содержащий кнопки Шаг, Прыжок, Поворот, Выполнить, Остановить, Проверить, Ручное выполнение, Автоматическое выполнение, Выход.

- 3) **Область редактирования** программ.
- 4) **Рабочая среда** Кенгуру, которая представляет собой прямоугольное поле, разлинованное на клетки.

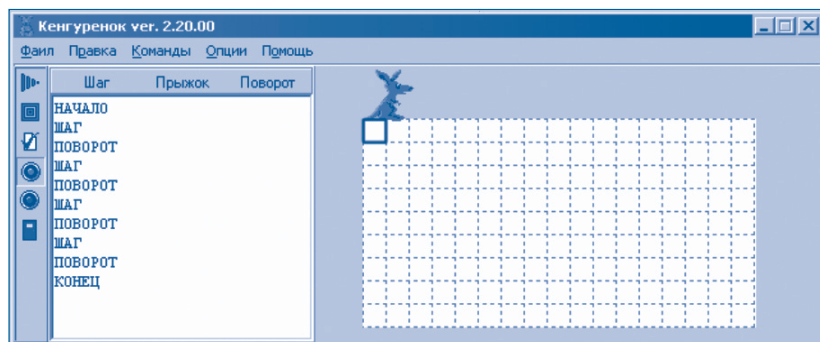


Рис. 2.1. Окно приложения **Кенгуренок**

Рассмотрим два режима управления исполнителями: ручное управление и программное управление.

**Режим ручного управления предполагает отдельный ввод каждой команды и немедленное ее выполнение исполнителем.**

Например, предположим, что мы хотим нарисовать квадрат определенных размеров. В случае ручного управления пользователь должен нажимать кнопки **ШАГ**, **ПОВОРОТ** и **ПРЫЖОК** панели управления таким образом, чтобы **Кенгуренок** рисовал соответствующий квадрат.

**Режим программного управления предполагает предварительное сохранение определенной последовательности команд и их выполнение в автоматическом режиме, без вмешательства пользователя.**

В случае программного управления пользователь должен ввести в область редактирования программ последовательность предписаний (команд), которые обеспечивают рисование квадрата (рис. 2.1). Отметим, что начало и конец каждой программы указываются с помощью вспомогательных слов **НАЧАЛО** и **КОНЕЦ**.

**Программа представляет собой алгоритм, записанный на языке исполнителя. Процесс написания (разработки) программ называется программированием.**

Очевидно, что программы можно сохранять на диске и использовать их многократно. В случае приложения **Кенгуренок** сохранение и открытие программ осуществляются с помощью команд, сгруппированных в меню **Файл**.

Программы для исполнителей пишутся с помощью специальных языков, называемых **языками программирования**. Эти языки содержат предписания, называемые командами, и вспомогательные слова. Обычно команды исполнителя являются одновременно и командами языка программирования. Например, язык программирования исполнителя **Кенгуренок** содержит команды **ШАГ**, **ПРЫЖОК**, **ПОВОРОТ** и вспомогательные слова **НАЧАЛО** и **КОНЕЦ**.

## Исполнитель Муравей

В этом приложении исполнитель представлен изображением муравья, который может перемещаться по полю, разлинованному на клетки (рис. 2.2). В некоторые клетки могут быть вписаны печатные символы. Обычно требуется разработать программу, которая располагает эти символы в определенном порядке.

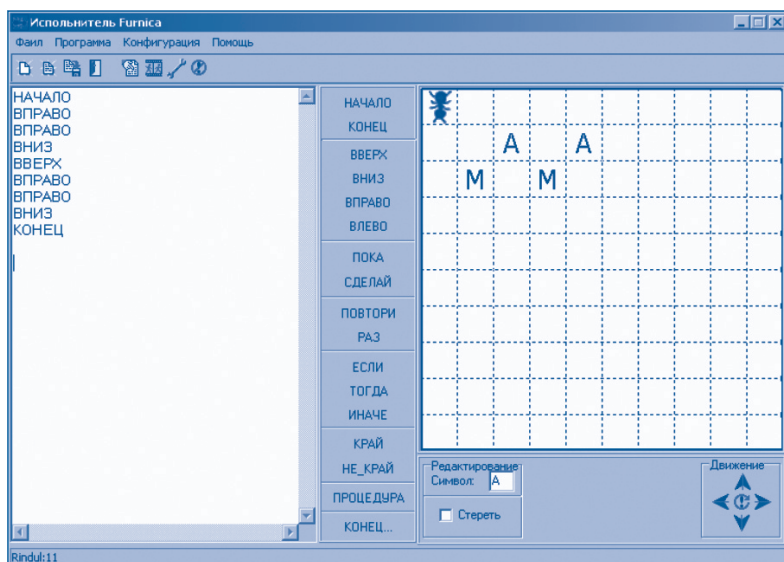


Рис. 2.2. Окно приложения **Муравей**

Исполнитель **Муравей** может выполнять команды ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, которые перемещают муравья из текущей в одну из соседних клеток. Если в соответствующей клетке находится какой-либо символ, он будет сдвинут, когда это возможно, по направлению движения. Начало и конец программы указываются с помощью вспомогательных слов НАЧАЛО и КОНЕЦ.

Окно приложения **Муравей** (рис. 2.2) содержит следующие элементы:

- 1) **Строка меню**, содержащая стандартные пункты меню Файл, Программа, Конфигурация, Помощь.
- 2) **Кнопки**: Новый, Открыть, Сохранить, Выход, Синтаксический анализ, Выполнить, Остановить.
- 3) **Область редактирования программ**.
- 4) **Рабочая среда** Муравья, которая представляет собой прямоугольное поле, разлинованное на клетки.
- 5) **Центр управления**, содержащий кнопки ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, обозначенные стрелками.
- 6) **Поле вставки** печатных символов.
- 7) **Кнопки**: НАЧАЛО, КОНЕЦ, ВВЕРХ, ВНИЗ, ..., ПРОЦЕДУРА, предназначенные для упрощения процесса редактирования программ.

В режиме ручного управления пользователь нажимает стрелки, обозначающие команды ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО. В режиме программного управления пользователь сначала вводит соответствующие команды в область редактирования. После ввода и/или редактирования программа может быть

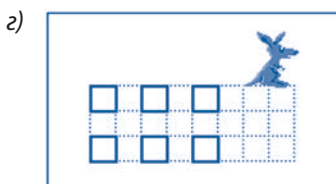
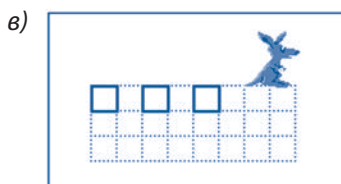
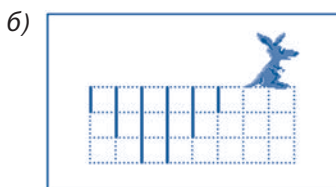
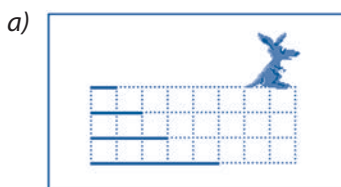


запущена на выполнение и/или сохранена на диске. Для примера на *рисунке 2.2* представлена программа, которая упорядочивает буквы из ячеек рабочей среды муравья таким образом, чтобы они образовали слово МАМА.

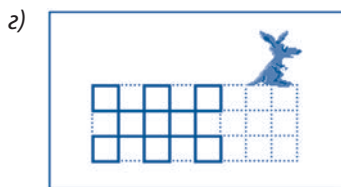
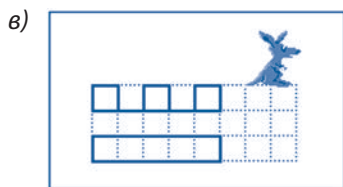
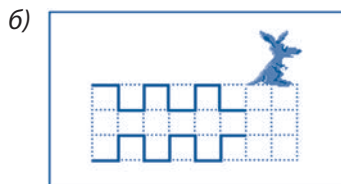
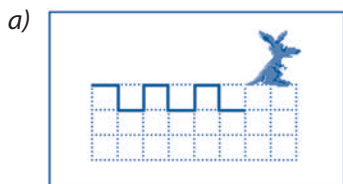
Отметим, что, независимо от типа исполнителя, программы играют роль «исходных данных» для центра управления, который в ходе выполнения программы анализирует каждую команду и генерирует соответствующие команды для исполнителя. В следующих параграфах мы увидим, что языки программирования содержат специальные команды, которые могут генерировать сотни и тысячи команд, описывая в компактной форме очень сложные алгоритмы обработки информации.

## Вопросы и упражнения

- ❶ Назовите хотя бы три алгоритма, которые вы изучали на уроках математики.
- ❷ Как вы думаете, можно ли считать алгоритмами кулинарные рецепты? Обоснуйте свой ответ.
- ❸ **СОЗДАЙТЕ!** Попробуйте сформулировать алгоритм, предназначенный для суммирования, вычитания, умножения и деления десятичных чисел. Сформулируйте подобные алгоритмы для двоичных чисел.
- ❹ Попробуйте составить полный список команд, которые может выполнять ваш телевизор.
- ❺ Составьте полный список команд, которые может выполнять ваш *player* (устройство для воспроизведения записей).
- ❻ Какие виды информации должно содержать полное описание каждого исполнителя?
- ❼ Опишите систему команд и рабочую среду исполнителя **Кенгуренок**.
- ❽ Опишите систему команд и рабочую среду исполнителя **Муравей**.
- ❾ В чем состоит разница между алгоритмами и программами? Обоснуйте свой ответ.
- ❿ Чем отличаются программный и ручной режимы управления исполнителем?
- ⓫ **ИССЛЕДУЙТЕ!** Запустите на выполнение приложение **Кенгуренок**. Найдите с помощью системы подсказки назначение всех элементов управления окна этого приложения.
- ⓬ **СОЗДАЙТЕ!** Выполняя команды ШАГ, ПРЫЖОК, ПОВОРОТ, нарисуйте приведенные ниже фигуры:



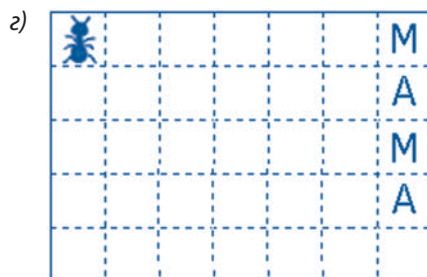
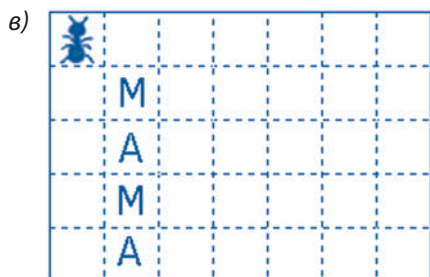
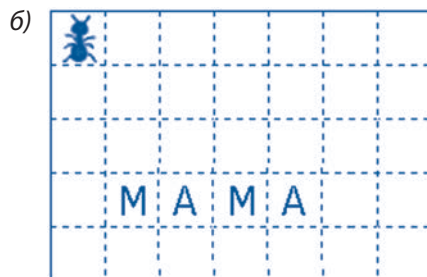
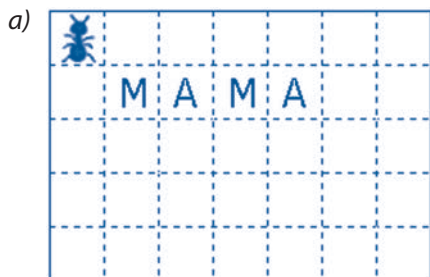
13 Напишите программы для рисования приведенных ниже фигур:



Сохраните эти программы на внешнем запоминающем устройстве.

14 **ИССЛЕДУЙТЕ!** Запустите на выполнение приложение **Муравей**. Найдите с помощью системы подсказки назначение всех элементов управления окна этого приложения.

15 **СОЗДАЙТЕ!** Выполняя команды **ВВЕРХ**, **ВНИЗ**, **ВПРАВО**, **ВЛЕВО**, упорядочьте символы рисунка 2.2 по приведенным ниже образцам:



16 **РАЗРАБОТАЙТЕ!** Напишите программы, предназначенные для упорядочения символов на рисунке 2.2 в соответствии с образцами, представленными в упражнении 15. Сохраните эти программы на магнитном диске.

17 Изобразите на рисунке фигуру, которую нарисует исполнитель **Кенгуренок** в результате выполнения приведенной ниже программы:

1  
НАЧАЛО  
ШАГ

2  
ШАГ  
ПОВОРОТ

3  
ШАГ  
ШАГ



ШАГ	ШАГ	ПОВОРОТ
ПОВОРОТ	ШАГ	КОНЕЦ
ШАГ	ПОВОРОТ	

- 18 Изобразите на рисунке размещение букв рисунка 2.2 после выполнения следующей программы:

<b>1</b>	<b>2</b>	<b>3</b>
НАЧАЛО	ВНИЗ	ВЛЕВО
ВНИЗ	ВПРАВО	ВЛЕВО
ВПРАВО	ВПРАВО	ВЛЕВО
ВПРАВО	ВВЕРХ	ВВЕРХ
ВЛЕВО	ВПРАВО	ВПРАВО
ВЛЕВО	ВВЕРХ	ВПРАВО
ВНИЗ	ВНИЗ	КОНЕЦ

- 19 Кто дает команды исполнителю в режиме ручного управления? А в режиме программного управления?
- 20 **УЧИТЕСЬ УЧИТЬСЯ!** Существует множество визуальных платформ и приложений с интуитивно понятным и простым в использовании интерфейсом, которые помогают ученикам легче изучать алгоритмы и программирование. Например, *Code.org*, *Lightbot*, *Scratch* и т. д. Изучите одно из таких приложений индивидуально или в сотрудничестве с одноклассниками, разработайте несколько проектов и представьте их другим ученикам в классе.

## 2.2. Субалгоритмы

*Ключевые термины:* • подпрограмма • главная программа  
• процедура • вызов процедуры (или обращение к процедуре)  
• метод последовательных уточнений

Как и в случае языков, используемых людьми, язык программирования описывается его синтаксисом и семантикой. Известно, что синтаксис – это набор правил, которые описывают структуру предложений, а семантика – это набор правил, описывающих смысл соответствующих предложений. Отметим, что в случае языков программирования эквивалентом предложения является программа.

Синтаксис программ может быть описан с помощью **форматов**, которые представляют собой образцы (модели, шаблоны) для написания команд, символов и вспомогательных слов. Например, для исполнителей **Кенгуренок** и **Муравей** программы, которые приводились выше, имели следующий формат:

```
НАЧАЛО
Команда_1
Команда_2
...
Команда_k
КОНЕЦ
```

где Команда\_к,  $k = 1, 2, 3$  и т.д. может быть любой из команд ШАГ, ПРЫЖОК, ПОВОРОТ для исполнителя **Кенгуренок** и ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО – для исполнителя **Муравей**.

Анализ многих практически важных задач позволяет выявить тот факт, что большинство из них содержат последовательности команд, выполняющих одни и те же операции (действия). Например, предположим, что необходимо нарисовать восемь квадратов, как это показано на рисунке 2.3.

Решая эту задачу «в лоб», можно написать программу, в которой каждый квадрат будет получен путем рисования всех сторон с помощью команд ШАГ и ПОВОРОТ. Очевидно, что такая программа получится очень длинной, а ее написание будет очень утомительным. Более

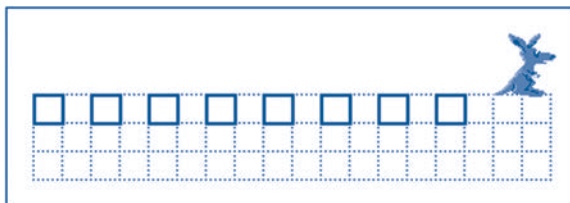


Рис. 2.3. Рисование квадратов

элегантное решение состоит в написании вспомогательной программы для рисования квадрата требуемых размеров и последующем использовании этой программы для рисования произвольного числа квадратов. Такая вспомогательная программа называется **подпрограммой**, а программа, которая к ней обращается (еще можно сказать – «ее вызывает»), – **главной программой**, или просто программой.

В случае исполнителей **Кенгуренок** и **Муравей** подпрограммы называются процедурами и имеют следующий формат:

```
ПРОЦЕДУРА Имя
Команда_1
Команда_2
...
Команда_k
КОНЕЦ ПРОЦЕДУРЫ
```

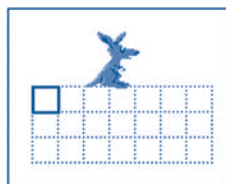
Имя процедуры представляет собой последовательность, состоящую из букв и цифр, начинающуюся с буквы. Слово ПРОЦЕДУРА – не команда, а вспомогательное слово, которое сообщает центру управления имя процедуры и указывает место в тексте программы, где начинается ее описание. Окончание описания процедуры указано с помощью вспомогательных слов КОНЕЦ ПРОЦЕДУРЫ. Команды, расположенные между вспомогательными словами ПРОЦЕДУРА и КОНЕЦ ПРОЦЕДУРЫ, образуют **тело процедуры**.

Пример:

**Процедура Квадрат**

```
ПРОЦЕДУРА
Квадрат
ШАГ
ПОВОРОТ
ШАГ
ПОВОРОТ
ШАГ
```

**Получаемая фигура**



```
ПОВОРОТ
ШАГ
ПОВОРОТ
ПРЫЖОК
ПРЫЖОК
КОНЕЦ ПРОЦЕДУРЫ
```

Процедура Квадрат рисует квадрат с начальным положением Кенгуренка в верхнем левом углу. Конечное положение Кенгуренка было выбрано таким образом, чтобы оно совпадало с начальным положением следующего квадрата (рис. 2.3).

Команды, входящие в состав процедур, будут выполняться только тогда, когда в ходе выполнения главной программы встретится команда **вызова процедуры**. Эта команда имеет следующий формат:

```
ВЫПОЛНИТЬ Имя
```

где слово *Имя* представляет собой имя вызываемой процедуры.

Когда Центр управления встречает команду вызова процедуры, выполнение главной программы приостанавливается и начинается выполнение команд, входящих в состав вызываемой процедуры. После завершения выполнения команд процедуры Центр управления возвращается к выполнению главной программы, а именно к команде следующей за командой вызова соответствующей процедуры. Следовательно, программа Восемь квадратов, предназначенная для рисования квадратов *рисунка 2.3*, имеет вид:

```
НАЧАЛО
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
ВЫПОЛНИТЬ Квадрат
КОНЕЦ
```

Очевидно, что главной программе должны быть известны описания всех процедур, которые будут в ней вызываться. В зависимости от типа исполнителя, соответствующие процедуры могут быть включены в основную программу или записаны в отдельных файлах на внешних запоминающих устройствах.

Для исполнителей **Кенгуренок** и **Муравей** процедуры включаются в начало основной программы. Общий формат программы имеет вид:

```
ПРОЦЕДУРА Имя_1
...
КОНЕЦ ПРОЦЕДУРЫ
ПРОЦЕДУРА Имя_2
...
КОНЕЦ ПРОЦЕДУРЫ
...
```

```
НАЧАЛО  
Команда_1  
Команда_2  
...  
Команда_k  
КОНЕЦ
```

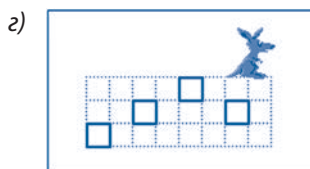
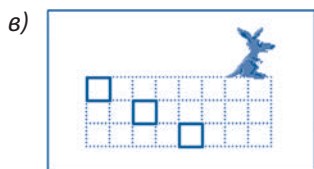
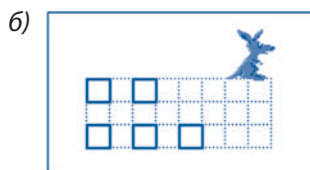
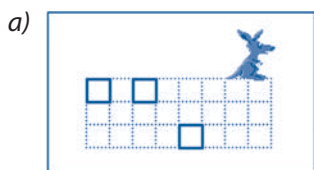
Из представленного формата видим, что программа состоит из описания подпрограмм *Имя\_1*, *Имя\_2* и т.д., за которыми следуют команды главной программы, заключенные между вспомогательными словами НАЧАЛО и КОНЕЦ. Вспомогательные слова и команды, заключенные между ними, образуют тело программы. Следовательно, программа состоит из **описаний подпрограмм** и собственно **тела программы**.

Вспомним, что программа представляет собой алгоритм, записанный на языке исполнителя. Ясно, что классификация программ на подпрограммы и главные программы применима и для алгоритмов. Как правило, для решения определенной задачи пользователь разрабатывает требуемые алгоритмы и субалгоритмы, создавая соответствующие программы и подпрограммы для каждого конкретного типа исполнителя.

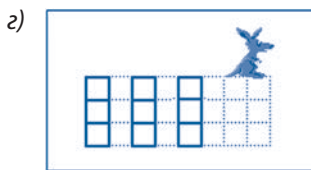
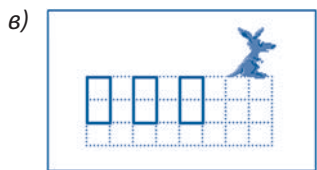
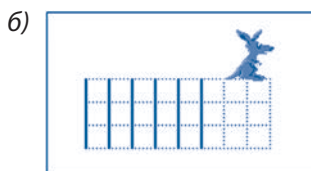
Подпрограммы представляют собой субалгоритмы, предназначенные для решения отдельных задач, часто встречающихся при разработке главного алгоритма. Метод решения сложных задач путем разделения их на более простые задачи называется **методом последовательных уточнений**. Например, в случае *рисунка 2.3* начальная задача – рисование восьми квадратов – была разделена на восемь идентичных подзадач, а именно: рисование одного квадрата.

## Вопросы и упражнения

- ❶ Когда появляется необходимость использования субалгоритмов? Приведите примеры.
- ❷ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** В чем разница между алгоритмом и субалгоритмом?
- ❸ Каков формат процедур, записанных на языке исполнителей **Кенгуренок** и **Муравей**? Как вызываются эти процедуры?
- ❹ **РАЗРАБОТАЙТЕ!** Используя процедуру **Квадрат**, разработайте программы для рисования следующих фигур:



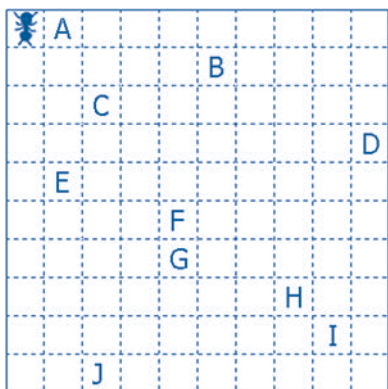
- 6 **АНАЛИЗИРУЙТЕ!** Найдите на приведенных рисунках повторяющиеся фрагменты. Напишите процедуры для рисования этих фрагментов.



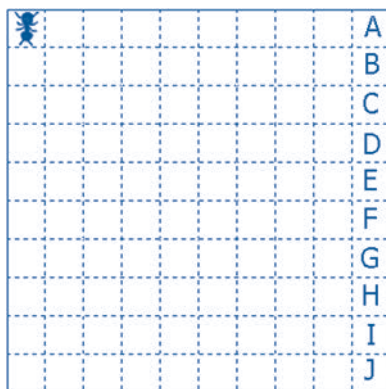
Используя разработанные процедуры, напишите программы для рисования представленных на рисунках фигур.

- 6 Используя метод последовательных уточнений, напишите программы рисования фигур в и г из упражнения 12 параграфа 2.1.
- 7 Напишите на языке исполнителя **Муравей** следующие процедуры:  
 Влево\_5 – перемещение Муравья влево на пять позиций;  
 Вправо\_5 – перемещение Муравья вправо на пять позиций;  
 Вверх\_5 – перемещение Муравья вверх на пять позиций;  
 Вниз\_5 – перемещение Муравья вниз на пять позиций.  
 Используя разработанные процедуры, напишите программу, которая перемещает Муравья таким образом, чтобы траектория движения представляла собой квадрат со сторонами а)  $5 \times 5$ ; б)  $6 \times 6$ ; в)  $8 \times 8$ .
- 8 На приведенных ниже фигурах представлен начальный и конечный вид рабочего поля исполнителя **Муравей**.

Начальный вид рабочего поля



Конечный вид рабочего поля



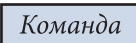
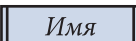


Напишите процедуру, которая перемещает печатный символ текущей строки в последнюю ячейку этой же строки. Напишите программу, которая перемещает все символы в последний столбец поля.

## 2.3. Циклические алгоритмы. Цикл со счетчиком

*Ключевые термины:* • блок-схема • линейный алгоритм • составная команда • простая команда • цикл со счетчиком • циклический алгоритм

С целью наглядного представления алгоритмы могут быть описаны с помощью блок-схем. **Блок-схема** представляет собой рисунок, состоящий из следующих графических обозначений:

	- точка запуска процесса выполнения алгоритма;
	- точка остановки процесса выполнения алгоритма;
	- выполнение указанной команды;
	- вызов указанного субалгоритма;

→ стрелка, указывающая порядок, в котором должны выполняться команды алгоритма.

В качестве примера на *рисунке 2.4* представлена блок-схема процедуры Квадрат и блок-схема программы Восемь\_квадратов.

Из анализа *рисунка 2.4* видим, что процесс выполнения алгоритма можно представить в виде воображаемого перехода от одного графического символа блок-схемы к другому в направлении, указанном стрелками.

**Алгоритмы, команды которых выполняются в порядке их появления в тексте, называются линейными.**

Для линейных алгоритмов воображаемый путь от графического символа СТАРТ до графического символа СТОП представляет собой линию без самопересечений (*рис. 2.4*).

В процессе разработки алгоритмов было установлено, что очень часто определенные последовательности команд должны выполняться многократно. Например, в случае процедуры Квадрат (*рис. 2.4 а*) последовательность команд ШАГ, ПОВОРОТ выполняется четыре раза, а команда вызова процедуры в программе Восемь\_квадратов – восемь раз. Для упрощения процесса разработки алгоритмов можно воспользоваться командой ПОВТОРИ. Формат и блок-схема названной команды представлены на *рисунке 2.5*.

В описании команды ПОВТОРИ *n* представляет число повторений, а ПОВТОРИ, РАЗ, КОНЕЦ ПОВТОРА являются вспомогательными словами.

Команда ПОВТОРИ записывается в нескольких строках и включает в себя другие команды. Команды такого вида называются составными командами в отличие от **простых команд** ШАГ, ПРЫЖОК, ПОВОРОТ, ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, вызов процедуры, рассмотренных в предыдущих параграфах.

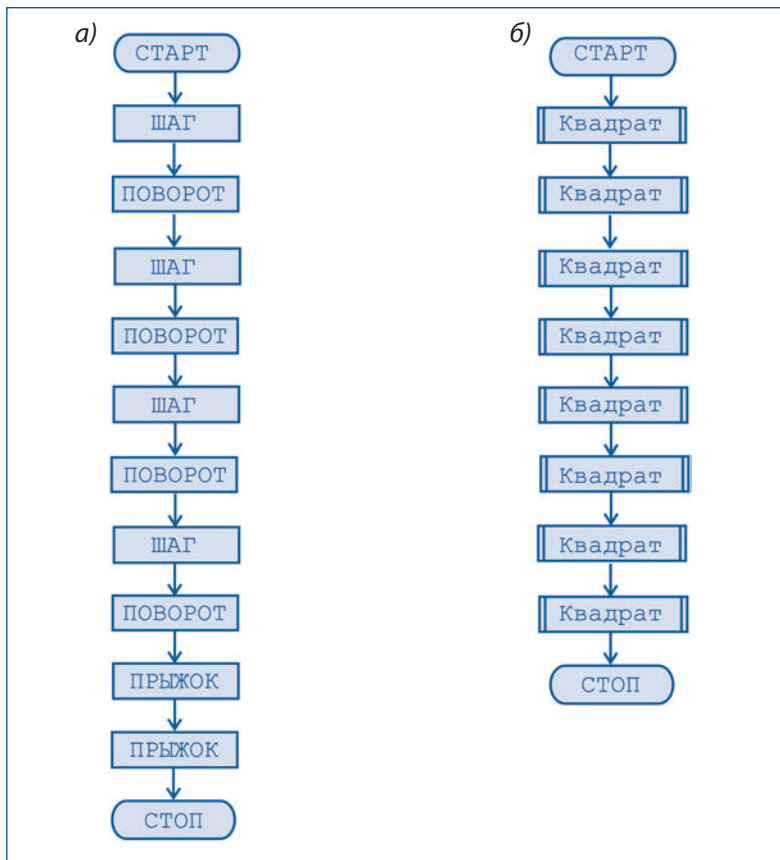


Рис. 2.4. Блок-схемы:  
 а – процедуры Квадрат; б – программы Восемь\_квадратов

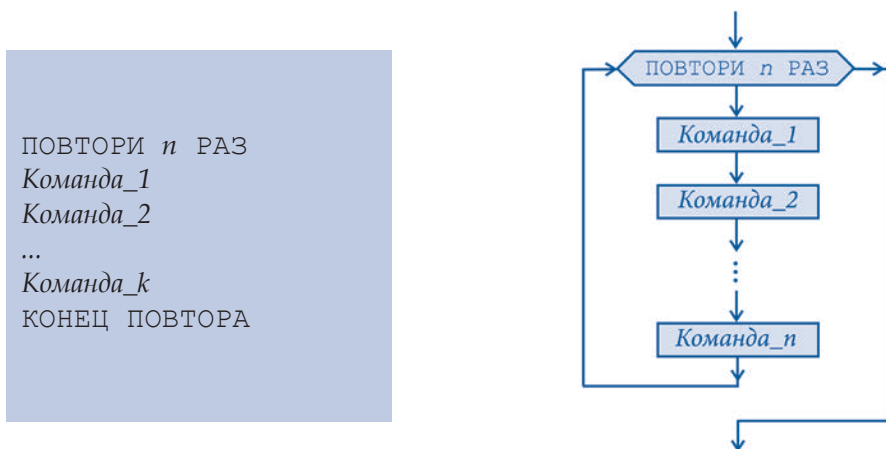


Рис. 2.5. Формат и блок-схема команды ПОВТОРИ

В ходе выполнения команды ПОВТОРИ Центр управления выполняет  $n$  раз группу команд, расположенных между соответствующими вспомогательными

словами. С помощью команды ПОВТОРИ программы, содержащие последовательности многократно повторяемых команд, можно переписать в более компактной форме. Например, программа Восемь\_квадратов может быть переписана в следующем виде:

1

```
ПРОЦЕДУРА Квадрат
ПОВТОРИ 4 РАЗ
ШАГ
ПОВОРОТ
КОНЕЦ ПОВТОРА
ПРЫЖОК
ПРЫЖОК
КОНЕЦ ПРОЦЕДУРЫ
```

2

```
НАЧАЛО
ПОВТОРИ 8 РАЗ
ВЫПОЛНИТЬ Квадрат
КОНЕЦ ПОВТОРА
КОНЕЦ
```

ПОВТОРИ и РАЗ называется **цикл со счетчиком**, поскольку при ее выполнении одна и та же последовательность команд повторяется многократно, а само число повторений  $n$  известно в момент написания программы. Группа команд, заключенная между вспомогательными словами ПОВТОРИ и КОНЕЦ ПОВТОРА, называется **телом цикла**.

**Алгоритмы, которые содержат последовательности многократно выполняемых команд, называются циклическими.**

Блок-схемы, иллюстрирующие в графическом виде процессы выполнения процедуры Квадрат и программы Восемь\_квадратов, представлены на рисунке 2.6. В этих блок-схемах используется графический символ ПОВТОРИ, из которого, в отличие от ранее изученных символов, выходят две стрелки: первая направлена к командам тела цикла, а вторая – к команде, которая будет выполняться непосредственно после окончания цикла.

Из анализа блок-схем можно заметить, что в случае циклических алгоритмов воображаемый путь от символа СТАРТ до символа СТОП представляет собой линию, содержащую как минимум одну петлю. Эта петля включает в себя графический символ ПОВТОРИ и все символы, соответствующие командам тела цикла (рис. 2.6).

Команда ПОВТОРИ может содержать в своем теле другие составные команды, образуя, таким образом, вложенную структуру. Следовательно, относительно короткие программы могут описывать очень длинные последовательности требуемых операций. В качестве примера приведем программу, которая заставляет **Кенгуренка** переместиться 100 раз вдоль верхнего края рабочего поля:

```
НАЧАЛО
ПОВТОРИ 100 РАЗ
ПОВТОРИ 15 РАЗ
ПРЫЖОК
КОНЕЦ ПОВТОРА
```



ПОВОРОТ  
ПОВОРОТ  
КОНЕЦ ПОВТОРА  
КОНЕЦ

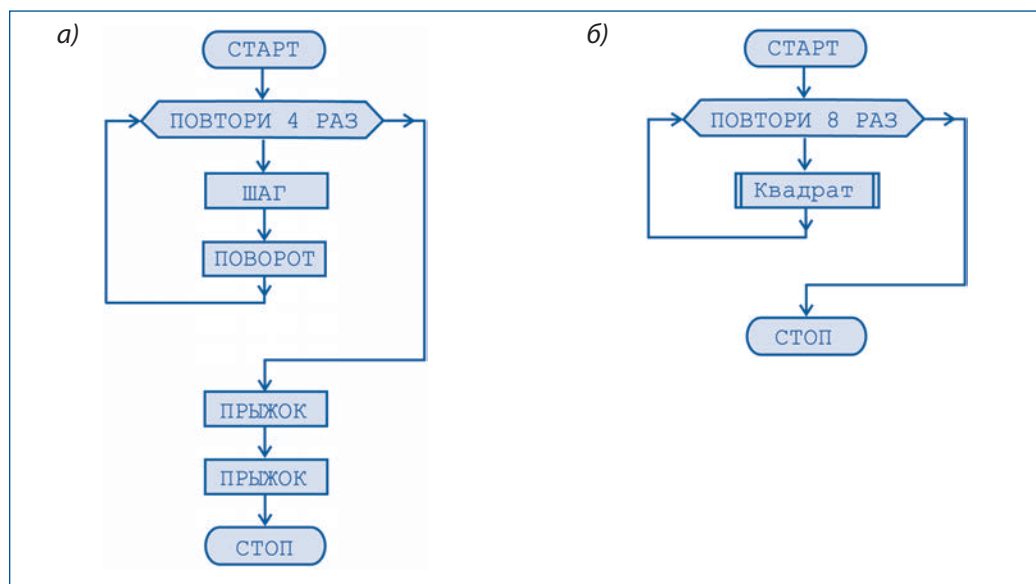


Рис. 2.6. Блок-схемы циклических алгоритмов:  
а – процедуры Квадрат; б – программы Восемь\_квадратов

## Вопросы и упражнения

❶ Объясните смысл следующих графических элементов:

а)

в)

д)

б)

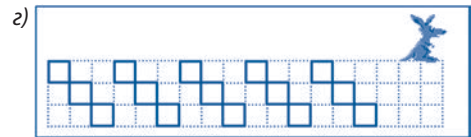
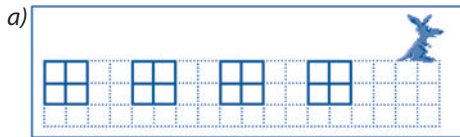
г)

е) →

Каков смысл стрелок в блок-схемах?

- ❷ **СОЗДАЙТЕ!** Нарисуйте блок-схемы алгоритмов рисования фигур из упражнений 4, 5 и 8 из параграфа 2.2.
- ❸ Покажите на блок-схемах рисунка 2.4 воображаемый путь, представляющий процесс выполнения соответствующих алгоритмов.
- ❹ Объясните термин линейный алгоритм. Приведите примеры.
- ❺ Каков формат команды ПОВТОРИ? Приведите несколько примеров использования этой команды.
- ❻ Как вы думаете, когда применяется команда ПОВТОРИ? Приведите примеры.
- ❼ Объясните термин циклический алгоритм. Приведите примеры.
- ❽ Покажите на блок-схемах рисунка 2.6 воображаемый путь, представляющий процесс выполнения соответствующих алгоритмов.

- 9 Покажите на блок-схемах рисунка 2.6 петли, представляющие циклы со счетчиком.
- 10 **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Перечислите известные вам простые команды. В чем отличие простой команды от составной команды?
- 11 Разработайте циклические алгоритмы для рисования следующих фигур:



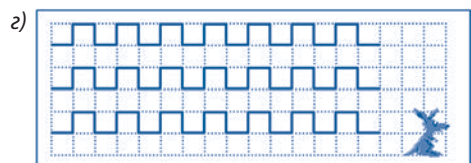
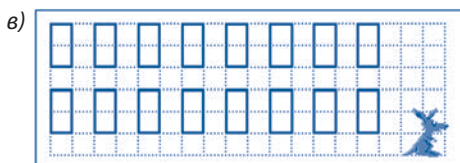
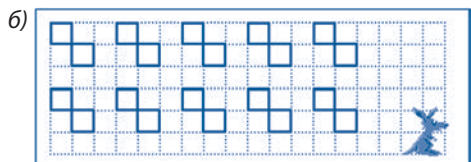
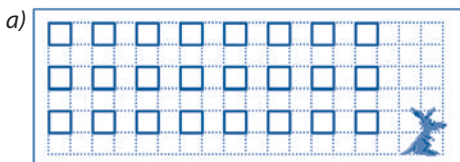
Подсчитайте, сколько команд выполнит **Кенгуренок** в процессе рисования этих фигур.

- 12 **АНАЛИЗИРУЙТЕ!** Сколько команд выполнит **Кенгуренок** в ходе выполнения следующих программ:

а) НАЧАЛО  
ПОВТОРИ 10 РАЗ  
ПРЫЖОК  
ПОВОРОТ  
ПОВОРОТ  
ПРЫЖОК  
КОНЕЦ ПОВТОРА  
КОНЕЦ

б) НАЧАЛО  
ПОВТОРИ 10 РАЗ  
ПОВТОРИ 20 РАЗ  
ПРЫЖОК  
ПОВОРОТ  
ПОВОРОТ  
ПРЫЖОК  
КОНЕЦ ПОВТОРА  
КОНЕЦ ПОВТОРА  
КОНЕЦ

- 13 Используя вложенные циклы, разработайте циклические алгоритмы для рисования следующих фигур:



Подсчитайте, сколько команд выполнит **Кенгуренок** при рисовании этих фигур.

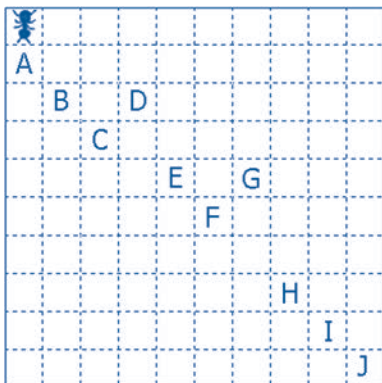
- 14 Запишите на языке исполнителя **Муравей** следующие циклические субалгоритмы:

Влево\_8 – перемещает Муравья влево на восемь позиций;  
Вправо\_8 – перемещает Муравья вправо на восемь позиций;  
Вверх\_8 – перемещает Муравья вверх на восемь позиций;  
Вниз\_8 – перемещает Муравья вниз на восемь позиций.

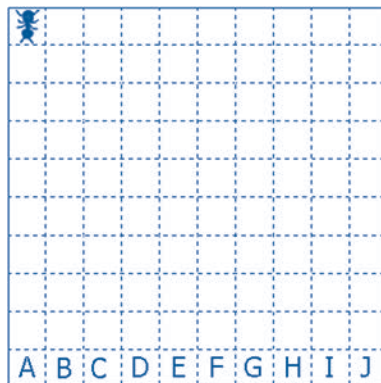
Используя эти субалгоритмы, напишите программу, которая перемещает Муравья таким образом, что траектория его движения будет представлять квадрат со сторонами: а)  $8 \times 8$ ; б)  $9 \times 9$ ; в)  $10 \times 10$ .

- 16 На приведенных ниже фигурах показаны начальный и конечный вид рабочего поля исполнителя **Муравей**. Используя циклы со счетчиком, напишите процедуру, которая перемещает печатный символ из текущего столбца в последнюю ячейку столбца.

Начальный вид рабочего поля



Конечный вид рабочего поля



Используя циклы со счетчиком, напишите программу, которая переместит все печатные символы в последнюю строку поля.

## 2.4. Циклические алгоритмы. Цикл с условием

*Ключевые термины:* • сенсор • условие • цикл с условием • алгоритм с обратной связью • ошибка выполнения (отказ)

До настоящего момента все алгоритмы, разработанные нами для исполнителей **Кенгуренок** и **Муравей**, не анализировали ситуацию рабочей среды, т.е. условия на рабочем поле. Другими словами, Центр управлял действиями исполнителей без проверки того факта, обеспечивают ли соответствующие команды достижение поставленной цели или в состоянии ли исполнитель выполнить полученные от Центра команды. Такой способ разработки алгоритмов существенно усложняет решение многих задач, встречающихся в повседневной практике.

Например, предположим, что начальное положение Муравья нам неизвестно. Требуется разработать алгоритм, который перемещает Муравья в верхний левый угол рабочего поля. Очевидно, что изученные нами ранее команды не позволяют решить рассматриваемую задачу, так как не ясно, сколько команд ВВЕРХ, ВЛЕВО должен получить исполнитель.

Для того чтобы избежать возникновения таких ситуаций, исполнители снабжены **сенсорами**<sup>1</sup>, которые передают в Центр управления определенную

<sup>1</sup> Сенсор – устройство, обнаруживающее определенное явление.

информацию о рабочей среде. Например, исполнители **Кенгуренок** и **Муравей** снабжены сенсорами, которые обнаруживают наличие препятствий по ходу предполагаемого движения. Данные от сенсоров передаются Центру управления посредством логических переменных **ЛИНИЯ** и **КРАЙ**, которые могут принимать значения **ЛОЖЬ** или **ИСТИНА**.

При использовании циклических алгоритмов анализ ситуации в рабочей среде осуществляется с помощью составной команды **ПОКА**. Эта команда обеспечивает циклическое выполнение группы команд до тех пор, пока ситуация в рабочей среде удовлетворяет определенным условиям. Формат и блок-схема команды **ПОКА** представлены на рисунке 2.7.

В описании команды **ПОКА** *Условие* является логическим выражением, а слова **ПОКА** и **КОНЕЦ ЦИКЛА** – вспомогательными словами.

**Условия представляют собой логические выражения, принимающие значение ИСТИНА при наличии определенных ситуаций в рабочей среде исполнителя.**

В языках программирования исполнителей **Кенгуренок** и **Муравей** логические выражения, используемые для выявления определенных ситуаций в рабочей среде, очень просты:

#### Кенгуренок

ЛИНИЯ  
КРАЙ  
НЕ ЛИНИЯ  
НЕ КРАЙ

ПОКА *условие*  
Команда\_1  
Команда\_2  
...  
Команда\_k  
КОНЕЦ ЦИКЛА

#### Муравей

КРАЙ  
НЕ КРАЙ

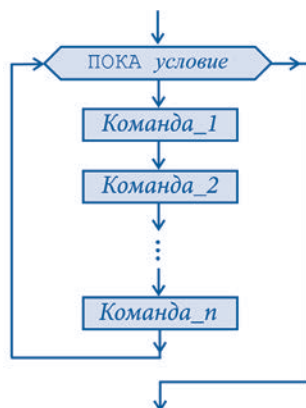


Рис. 2.7. Формат и блок-схема команды **ПОКА**

Для более сложных исполнителей в состав условий могут входить переменные, которые сообщают Центру управления информацию о направлении и скорости движения других объектов, наличии препятствий, температуре, влажности, уровне радиации и т.п., а логические выражения могут быть построены с помощью операторов отношения  $=$ ,  $\neq$ ,  $>$ ,  $\geq$ ,  $<$ ,  $\leq$  и логических операторов **НЕ**, **И**, **ИЛИ**.

Для примера рассмотрим следующую задачу. Предположим, что необходимо переместить **Кенгуренка** на один из краев рабочего поля, причем его

начальное положение неизвестно. Очевидно, что в данном случае решить задачу с помощью команды ПОВТОРИ невозможно, так как количество шагов, которые должен сделать Кенгуренок, заранее неизвестно. С помощью команды ПОКА рассматриваемая задача решается очень легко:

```
ПОКА НЕ КРАЙ  
ПРЫЖОК  
КОНЕЦ ЦИКЛА
```

В процессе выполнения команды ПОКА Центр управления, в первую очередь, вычисляет условие НЕ КРАЙ. Если указанное условие имеет значение ИСТИНА, выполняются команды тела цикла, в нашем случае – команда ПРЫЖОК. После выполнения команды ПРЫЖОК сенсор Кенгуренка исследует соседнюю клетку и обновляет значение переменной КРАЙ. Если условие НЕ КРАЙ принимает значение ЛОЖЬ, выполнение цикла завершается и осуществляется переход к команде, следующей непосредственно за вспомогательными словами КОНЕЦ ЦИКЛА.

Команда ПОКА называется **циклом с условием**, поскольку она обеспечивает многократное выполнение содержащейся в ней группы команд. Количество повторений устанавливается в ходе выполнения программы в зависимости от текущих значений указанного условия.

Реализация команды ПОКА предполагает существование между исполнителем и Центром управления двух каналов передачи информации:

- 1) прямого канала, предназначенного для передачи команд от Центра управления к исполнителю;
- 2) обратного канала, предназначенного для передачи информации, собранной сенсорами исполнителя, в Центр управления.

**Алгоритмы, содержащие группы команд, выполнение которых зависит от информации в рабочей среде исполнителя, называются алгоритмами с обратной связью.**

Отметим, что большинство алгоритмов, используемых в современной информатике, являются алгоритмами с обратной связью, что позволяет исполнителям адаптироваться (приспосабливаться) к конкретным ситуациям рабочей среды. В качестве примера рассмотрим очень маленькую программу, предназначенную для рисования спирали (рис. 2.8):

```
НАЧАЛО  
ПОКА НЕ ЛИНИЯ  
ПОКА НЕ ЛИНИЯ  
ШАГ  
КОНЕЦ ЦИКЛА  
ПОВОРОТ  
КОНЕЦ ЦИКЛА  
КОНЕЦ
```

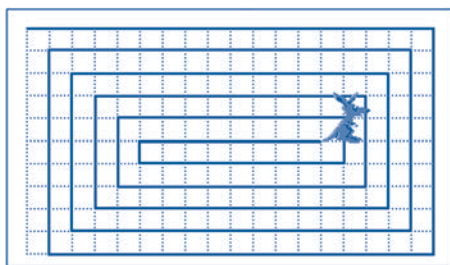


Рис. 2.8. Спираль, нарисованная Кенгуренком

Внутренний цикл, телом которого является команда ШАГ, рисует отрезок прямой, длина которого зависит от вида ранее нарисованных линий. Как только сенсор Кенгуренка обнаруживает в соседней клетке линию, условие НЕ ЛИНИЯ принимает значение ЛОЖЬ и выполнение этого цикла прекращается. Следующая команда поворачивает Кенгуренка на  $90^\circ$ , готовя его к проведению новой линии. Внешний цикл обеспечивает проведение линий, образующих спираль до тех пор, пока все соседние клетки окажутся занятыми.

Теоретически спираль можно нарисовать и без использования команды цикла ПОКА, создавая соответствующие циклы с помощью команды ПОВТОРИ. Однако в этом случае пользователь должен вычислить «вручную» длину каждой линии спирали и записать по четыре цикла со счетчиком для каждого витка. Очевидно, что при большом числе витков указанная работа становится очень утомительной или даже практически невозможной.

Из практики известно, что при разработке и отладке программ появляются ситуации, когда исполнителю посылаются такие команды, которые он не может выполнить. Например, программа

```
НАЧАЛО
ПОВТОРИ 1000 РАЗ
ШАГ
КОНЕЦ ПОВТОРА
КОНЕЦ
```

с синтаксической точки зрения является правильной, но ее полное выполнение невозможно, поскольку размеры рабочего поля значительно меньше, чем  $1000 \times 1000$ . В процессе выполнения рассматриваемой программы, независимо от исходного положения, возникает ситуация, при которой Кенгуренок не может в дальнейшем выполнять команду ШАГ, поскольку он уже находится на краю поля. Следовательно, в процессе выполнения приведенной выше программы появляется ошибка выполнения или, другими словами, отказ.

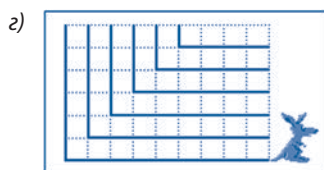
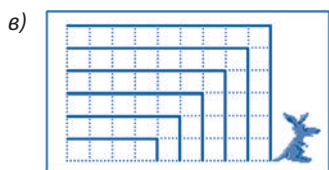
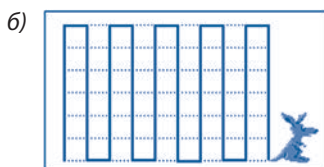
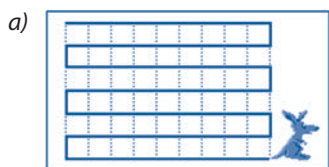
**Ошибка выполнения (отказ) появляется в том случае, когда в процессе выполнения программы исполнитель не может выполнить полученную команду.**

Реакция Центра управления на отказ зависит от типа исполнителя и возможностей соответствующей программы. Например, исполнители Кенгуренок и Муравей выводят сообщение об ошибке и останавливаются. В случае более сложных

исполнителей отказ понимается как особое условие, которое вызывает запуск специальных субалгоритмов для исправления возможных ошибок. В качестве примера вспомним системы подсказки текстовых редакторов или табличных процессоров, которые в случае возникновения ошибок выводят на экран подсказки, помогающие пользователю правильно вводить соответствующие команды.

## Вопросы и упражнения

- ❶ Для чего предназначены сенсоры, которыми снабжены исполнители?
- ❷ Для чего применяются условия? Откуда берутся текущие значения переменных, входящих в эти условия?
- ❸ **СОЗДАЙТЕ!** Изобразите на рисунке потоки информации между Центром управления и исполнителем. Для чего предназначены прямой и обратный каналы?
- ❹ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Чем отличаются алгоритмы с обратной связью от алгоритмов без такой связи?
- ❺ Как вы считаете, мог бы компьютер управлять автомобилем без использования алгоритмов с обратной связью? Обоснуйте свой ответ.
- ❻ **АНАЛИЗИРУЙТЕ!** Нарисуйте блок-схему программы для рисования спирали. Укажите на блок-схеме воображаемый путь, представляющий процесс выполнения программы, и петли, которые соответствуют циклам с условиями.
- ❼ **ИССЛЕДУЙТЕ!** Попробуйте разработать программу, которая рисует спираль без использования цикла с условием. Оцените (подсчитайте) количество команд такой программы.
- ❽ Используя циклы с условиями, напишите программы для рисования приведенных ниже фигур:



Нарисуйте блок-схемы разработанных вами программ. Укажите на каждой блок-схеме воображаемый путь, представляющий процесс выполнения программы, и петли, которые соответствуют циклам с условиями.

- ❾ Разработайте программу, которая независимо от начального положения Муравья перемещает его в одну из следующих позиций рабочего поля:
  - а) в верхний левый угол;
  - б) в верхний правый угол;
  - в) в левый нижний угол;
  - г) в правый нижний угол.
 Считается, что все клетки рабочего поля являются свободными.
- ❿ Разработайте программу, которая безостановочно перемещает Муравья таким образом, что траектория его движения представляет квадрат, стороны которого совпадают с краями рабочего поля. Начальное положение Муравья – левый верхний угол рабочего поля.



## 2.5. Алгоритмы с разветвлениями

*Ключевые термины:* • разветвление • алгоритм с разветвлениями

В процессе выполнения программ очень часто возникает необходимость задавать исполнителю команды в зависимости от ситуации в рабочей среде. Мы уже знаем, что информация о ситуации в рабочей среде собирается с помощью сенсоров, причем соответствующие данные представляются в программах в виде переменных **ЛИНИЯ**, **КРАЙ** и т.п. В предыдущем параграфе мы уже использовали эти переменные для записи условий в составных командах **ПОКА**.

Условия, которые описывают ситуацию в рабочей среде, могут быть проанализированы и с помощью другой составной команды оператора, а именно – команды **ЕСЛИ**. Формат и блок-схема команды **ЕСЛИ** представлены на рисунке 2.9.

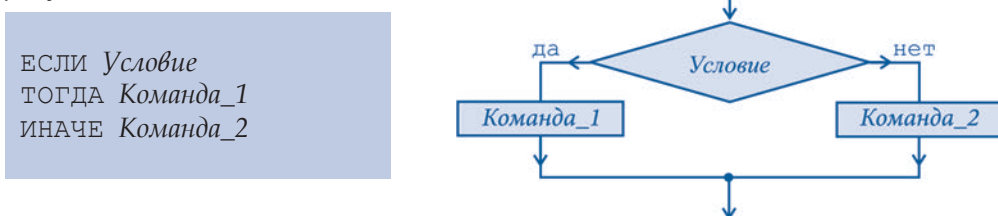


Рис. 2.9. Формат и блок-схема команды **ЕСЛИ**

В описании команды **ЕСЛИ** *Условие* – это логическое выражение, а **ЕСЛИ**, **ТОГДА** и **ИНАЧЕ** являются вспомогательными словами.

Как и в случае команды **ПОКА**, для исполнителей **Кенгуренок** и **Муравей** можно использовать следующие логические выражения:

ЛИНИЯ  
КРАЙ  
НЕ ЛИНИЯ  
НЕ КРАЙ

Очевидно, что в случае более сложных исполнителей в состав условий могут входить и другие переменные, а логические выражения могут записываться с использованием операторов отношения  $=$ ,  $\neq$ ,  $>$ ,  $\geq$ ,  $<$ ,  $\leq$  и логических операторов **НЕ**, **И**, **ИЛИ**.

В процессе выполнения команды **ЕСЛИ** Центр управления анализирует в первую очередь соответствующее условие. Если это условие имеет значение **ИСТИНА**, то выполняется *Команда\_1*, а в противном случае (условие имеет значение **ЛОЖЬ**) – выполняется *Команда\_2*.

Составная команда **ЕСЛИ** называется **командой разветвления**, так как воображаемый путь, представляющий процесс выполнения программы, проходит, в зависимости от текущего значения анализируемого условия, либо через



символ *Команда\_1*, либо через символ *Команда\_2*, а ромб представляет место разветвления.

**Алгоритмы, содержащие группы команд, которые выполняются только при определенных значениях заданных условий, называются алгоритмами с разветвлениями.**

В качестве примера рассмотрим программу, которая рисует квадрат, стороны которого совпадают с краями рабочего поля:

```
НАЧАЛО
ПОВТОРИ 100 РАЗ
ЕСЛИ КРАЙ
ТОГДА ПОВОРОТ
ИНАЧЕ ШАГ
КОНЕЦ ПОВТОРА
КОНЕЦ
```

Тело цикла ПОВТОРИ из приведенной программы содержит составную команду ЕСЛИ, которая будет выполнена 100 раз. При каждом выполнении команды ЕСЛИ проверяется условие КРАЙ и, в зависимости от его текущего значения, выполняется либо команда ПОВОРОТ, либо команда ШАГ. Очевидно, команда ПОВОРОТ выполняется только в тех случаях, когда условие принимает значение ИСТИНА, или, другими словами, когда Кенгуренок достигает края рабочего поля.

Блок-схема рассматриваемой программы представлена на *рисунке 2.10*. На блок-схеме видно, что при каждом повторном выполнении команды ЕСЛИ выбор одной из двух команд ПОВОРОТ или ШАГ осуществляется в зависимости от текущего положения Кенгуренка относительно края рабочего поля.

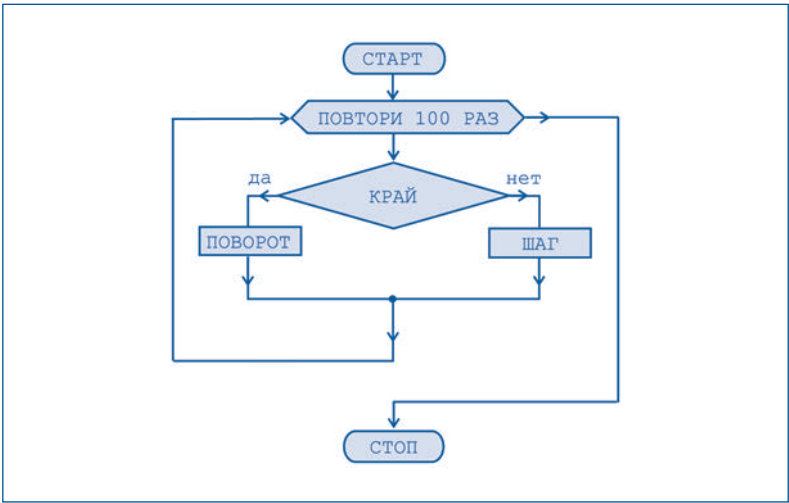


Рис. 2.10. Блок-схема алгоритма с разветвлением

## Вопросы и упражнения

- ❶ Для чего предназначена команда ЕСЛИ? Напишите формат и нарисуйте блок-схему этой команды.
- ❷ Как выполняется команда ЕСЛИ? Можно ли заменить команду ЕСЛИ командой ПОКА?
- ❸ Как Вы считаете, является ли алгоритм с разветвлениями алгоритмом с обратной связью? Обоснуйте свой ответ.
- ❹ **ИССЛЕДУЙТЕ!** Назовите как минимум три математические задачи, алгоритмы которых для решения содержат разветвления.
- ❺ В чем разница между линейными алгоритмами и алгоритмами с разветвлениями?
- ❻ В чем разница между циклическими алгоритмами и алгоритмами с разветвлениями?
- ❼ **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Напишите две программы, которые рисуют спираль внутри прямоугольника, размеры которого заранее не известны (рис. 2.11). Для анализа ситуации на рабочем поле в первой программе должна использоваться команда ЕСЛИ, а во второй – команда ПОКА. Как вы думаете, какая программа является более эффективной: та, что использует команду ЕСЛИ, или та, что использует команду ПОКА? Обоснуйте свой ответ.
- ❽ **СОЗДАЙТЕ!** Нарисуйте блок-схемы программ для рисования спирали внутри прямоугольника (рис. 2.11). Покажите на блок-схеме воображаемые пути, ход выполнения программы и точку разветвления.

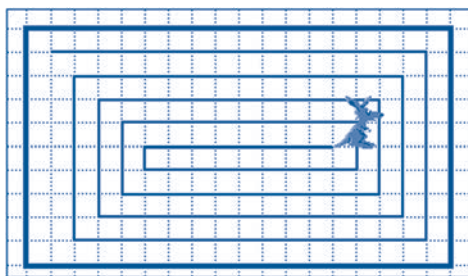


Рис. 2.11. Спираль внутри прямоугольника

- ❾ **РАЗРАБОТАЙТЕ!** Кенгуренок находится в начале коридора шириной в две клетки (рис. 2.12). Разработайте программу, которая рисует линию, проходящую точно по середине коридора. Предполагается, что остальные размеры коридора заранее неизвестны.

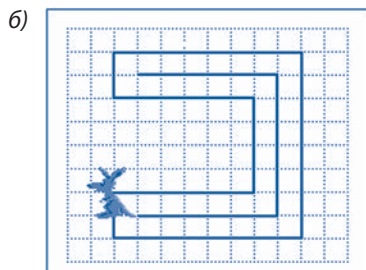
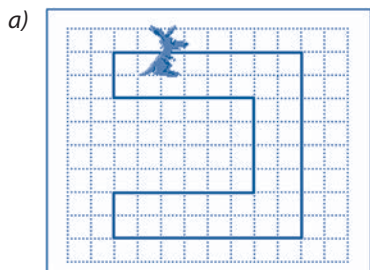


Рис. 2.12. Проведение линии внутри коридора  
а – начальное положение; б – конечное положение

## 2.6. Алгоритм работы компьютера

*Ключевые термины:* • функциональные блоки • центральное устройство управления • арифметико-логическое устройство • часто используемые команды • алгоритм работы компьютера

Любой компьютер состоит из следующих **функциональных блоков** (рис. 2.13): процессора, внутренней памяти, внешней памяти, устройства ввода и устройства вывода. Процессор, в свою очередь, состоит из **центрального устройства управления** и **арифметико-логического устройства**.

За исключением центрального устройства управления, все функциональные блоки компьютера могут рассматриваться как исполнители, выполняющие определенные наборы команд. В качестве примера в *таблице 2.1* представлены **часто используемые команды** исполнителей, которые входят в состав компьютера.

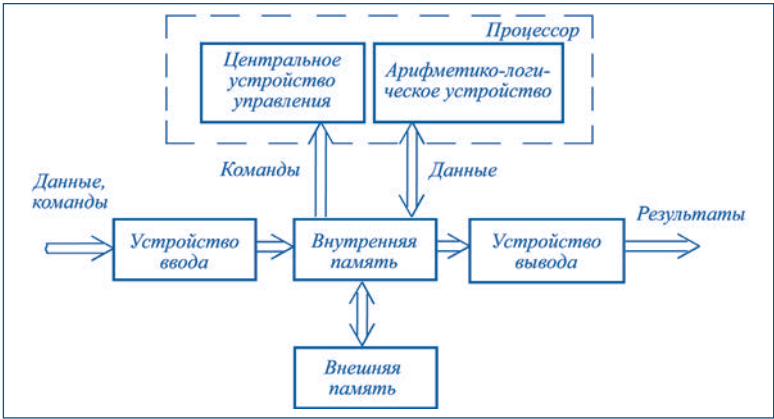


Рис. 2.13. Функциональная схема компьютера

Таблица 2.1

**Часто используемые команды исполнителей из состава персонального компьютера**

№	Исполнитель	Команды
1.	Клавиатура	Жди нажатия произвольной клавиши Прочти код нажатой клавиши Блокируй клавиатуру
2.	Монитор	Очисти экран Выведи указанный символ Выведи указанный графический элемент Установи цвет фона
3.	Устройство на магнитных дисках	Запиши данные на диск Прочитай данные с диска

4.	Принтер	Напечатай указанный символ Продвинь лист на строку Продвинь страницу
5.	Устройство на оптических дисках	Прочитай данные с диска Выдвинь (извлеки) диск
6.	Внутренняя память	Прочитай данные из указанной ячейки Запиши данные в указанную ячейку
7.	Арифметико-логическое устройство	Сложи Вычти Умножь Подели Сравни И ИЛИ

Центральное устройство управления отдает команды исполнителям в соответствии с программой, записанной во внутренней памяти компьютера. Программа состоит из набора машинных команд, в которых указываются операция, которую необходимо выполнить, и местоположение операндов. Команды закодированы с помощью двоичных чисел. Например, в любой арифметической команде указываются операция, которая должна быть выполнена (сложение, вычитание, умножение или деление), и местоположение операндов во внутренней памяти. В команде ввода информации указываются требуемое устройство ввода (клавиатура, сканнер) и место внутренней памяти, куда будет записана введенная информация. В команде вывода информации указываются место во внутренней памяти, содержащее извлекаемые данные, и требуемое устройство вывода (монитор или принтер).

Аналогичным образом, в случае работы с магнитным диском (чтение или запись информации), в соответствующей машинной команде указываются место размещения данных во внутренней памяти и требуемое дисковое устройство.

**Алгоритм работы** центрального устройства управления, а значит, и **компьютера** в целом может быть описан следующим образом:

**ПОКА** ИЗВЛЕКАЕМАЯ ИЗ ВНУТРЕННЕЙ ПАМЯТИ МАШИННАЯ КОМАНДА  $\neq$  СТОП  
 ИЗВЛЕКИ МАШИННУЮ КОМАНДУ ИЗ ВНУТРЕННЕЙ ПАМЯТИ  
 ДЕКОДИРУЙ ИЗВЛЕЧЕННУЮ МАШИННУЮ КОМАНДУ  
 ПЕРЕДАЙ КОМАНДЫ ИСПОЛНИТЕЛЯМ  
**КОНЕЦ ЦИКЛА**

Из приведенного алгоритма видно, что центральное устройство управления реализует принцип программного управления. Он состоит в том, что исполнителям передаются команды, получаемые из машинных команд, которые извлекаются из внутренней памяти компьютера. Подчеркнем, что система (набор) машинных команд современного компьютера включает в себя как команды обработки информации (сложение, вычитание, деление, умножение и т.п.), так и команды вызова подпрограмм, циклов и разветвления. Такой набор

дает возможность компактного написания очень сложных алгоритмов, которые при очень большом быстродействии (сотни миллионов операций в секунду) обеспечивают эффективное использование компьютеров во всех областях современной науки и техники.

## Вопросы и упражнения

- ❶ **ИССЛЕДУЙТЕ!** Определите набор команд каждого из периферийных устройств (клавиатуры, принтера, сканнера и т.п.), используемых в кабинете информатики.
- ❷ **РАЗРАБОТАЙТЕ!** Используя технические описания компьютеров из кабинета информатики, определите набор машинных команд компьютера, с которым вы работаете. Укажите команды обработки, команды ввода/вывода и команды управления.
- ❸ Объясните назначение каждого функционального блока, из которых состоит компьютер (рис. 2.13).
- ❹ Какую роль играет центральное устройство управления, входящее в состав процессора?
- ❺ Разработайте алгоритм функционирования цифрового компьютера. Как вы думаете, команды какого вида посылает центральное устройство управления исполнителям компьютера?
- ❻ **АНАЛИЗИРУЙТЕ!** От чего зависит производительность компьютера? Обоснуйте свой ответ.
- ❼ Какие действия совершает центральное устройство управления в процессе выполнения программ? Как можно прервать выполнение программы?
- ❽ Как реализуется принцип программного управления в случае цифровых компьютеров?
- ❾ **ИССЛЕДУЙТЕ!** Как вы думаете, для чего предназначены компьютеры, установленные в современных автомобилях? Какие виды периферийных устройств имеются у таких компьютеров? Какие программы выполняются на этих компьютерах?

## 2.7. Основные сведения об алгоритмах

*Ключевые термины:* • представление алгоритмов • классификация алгоритмов • свойства алгоритмов • алгоритмическое мышление • информационная культура

Для упрощения процесса разработки **алгоритмов** договорились, что они будут **представлены** (описаны, обозначены) с помощью специальных стандартных средств, самыми распространенными из которых являются:

- 1) обычные языки общения между людьми (например, русский) с использованием, при необходимости, математических формул;
- 2) блок-схемы;
- 3) алгоритмические языки (языки программирования).

Например, алгоритм решения уравнений первой степени  $ax + b = 0$  можно описать на русском языке следующим образом:

1. Прочитай с клавиатуры коэффициенты  $a$  и  $b$ .
2. Если  $a \neq 0$ , то вычисли корень  $x = -b/a$ , выведи на экран сообщение «Уравнение имеет один корень» и значение  $x$ . СТОП.
3. Если  $a = 0$  и  $b = 0$ , то выведи на экран сообщение «Уравнение имеет бесконечное множество корней». СТОП.
4. Если  $a = 0$  и  $b \neq 0$ , то выведи на экран сообщение «Уравнение не имеет смысла». СТОП.

Основное преимущество описания алгоритмов на обычном языке состоит в том, что такие описания являются понятными любому человеку, которому знакомы команды и действия, встречающиеся в тексте алгоритма. К сожалению, современные компьютеры не понимают однозначным образом языки, на которых общаются люди. Следовательно, алгоритмы, написанные на таких языках, не могут использоваться в качестве компьютерных программ.

В случае блок-схем алгоритмы описываются с помощью графических символов *старт*, *стоп*, *команда*, *вызов субалгоритма*, *повтори*, *пока*, *если* и др. В качестве примера на рисунке 2.14 представлена блок-схема алгоритма решения уравнений первой степени.

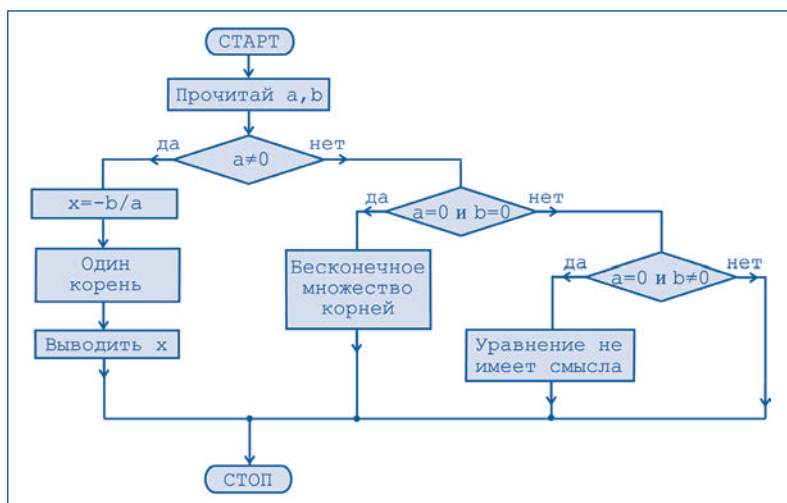


Рис. 2.14. Блок-схема алгоритма решения уравнений первой степени

Анализ блок-схем позволяет легче **классифицировать алгоритмы** на линейные, циклические и с разветвлениями. Блок-схемы очень наглядны, но, как и в случае языка общения между людьми, не могут однозначно восприниматься современными компьютерами. Как следствие, блок-схемы также не могут использоваться в качестве компьютерных программ.

Алгоритмические языки представляют собой искусственные языки, созданные с целью точного описания алгоритмов, и основаны на использовании строго

определенных словаря, синтаксиса и семантики. В качестве примера рассмотрим алгоритм решения уравнений первой степени, записанный на алгоритмическом языке ПАСКАЛЬ:

```
Program Exemplu;
var a, b, x : real;
begin
  readln(a, b);
  if a<>0 then
    begin
      x:=-b/a;
      writeln('Уравнение имеет один корень');
      writeln(x);
    end;
  if (a=0) and (b=0) then
    writeln('Уравнение имеет бесконечное множество корней');
  if (a=0) and (b<>0) then
    writeln('Уравнение не имеет смысла');
end.
```

Английские слова в этом описании имеют следующий смысл: **Program** – программа; **var** – переменная; **begin** – начало; **end** – конец; **if** – если; **then** – тогда; **readln** – прочитай; **writeln** – запиши; **and** – и.

Из приведенного примера видно, что алгоритмический язык ПАСКАЛЬ содержит команды и вспомогательные слова, изученные нами в предыдущих параграфах при разработке программ управления исполнителями **Кенгуренок** и **Муравей**.

Основным преимуществом алгоритмических языков является тот факт, что они обеспечивают однозначное описание алгоритмов, понятны людям и относительно легко переводятся на язык исполнителей. Очевидно изучение определенного алгоритмического языка, как и изучение любого иностранного языка, требует от человека определенных усилий.

В школах нашей страны алгоритмический язык ПАСКАЛЬ изучается начиная с 10 класса.

Языки программирования представляют собой алгоритмические языки, адаптированные (приспособленные) для описания алгоритмов в виде, понятном для исполнителей. Например, программы, разработанные в предыдущих параграфах, были написаны на языках исполнителей **Кенгуренок** и **Муравей**. Этот язык программирования содержит, кроме команд ПОВТОРИ, ПОКА, ЕСЛИ, вызов процедуры и др., встречающихся во всех алгоритмических языках, и команды, специфичные именно для данных исполнителей: ШАГ, ПРЫЖОК, ПОВОРОТ, ВВЕРХ, ВНИЗ и т.д.

Независимо от формы представления, **алгоритмы** имеют ряд общих **свойств**, которые отличают их от предписаний, рекомендаций или планов,



предназначенных для решения отдельных задач. Установлено, что любой алгоритм должен иметь следующие три свойства:

1) **однозначность** – при выполнении алгоритма на каждом шаге должно быть точно известно, как выполнить текущую операцию и какая именно операция будет следующей;

2) **универсальность** – алгоритм должен быть применим ко всем задачам, для которых он был разработан;

3) **конечность** – алгоритм конечен в пространстве (при описании) и во времени (при выполнении).

Обычно в ходе разработки пользователь делает набросок алгоритма, используя обычный язык, например русский. При этом для большей наглядности могут использоваться и блок-схемы. Безусловно, что в конечном виде алгоритм будет записан на определенном алгоритмическом языке или в случае конкретного исполнителя – непосредственно на соответствующем языке программирования.

Разработка алгоритмов представляет собой творческий процесс, предполагающий, что пользователь обладает так называемым алгоритмическим мышлением.

**Под алгоритмическим мышлением мы будем понимать способность человека разрабатывать алгоритмы для решения задач, с которыми он сталкивается в повседневной жизни.**

Безусловно, алгоритмическое мышление формируется и развивается в процессе изучения всех школьных дисциплин, но центральная роль при этом отводится информатике. Отметим, что если в прошлом веке главная задача учебных заведений заключалась в формировании навыков и привитии потребности в работе с книгами, то сегодня указанная задача значительно расширилась и включает в себя формирование **информационной культуры** и развитие алгоритмического мышления.

## Вопросы и упражнения

- 1 **ТЕМАТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Каковы основные средства представления алгоритмов? Какими преимуществами и недостатками обладает каждый из них?
- 2 Попробуйте описать процесс разработки произвольного алгоритма. Какие средства используются при описании алгоритма в ходе его разработки?
- 3 Существует ли разница между алгоритмическим языком и языком программирования? Аргументируйте свой ответ.
- 4 **ИССЛЕДУЙТЕ!** Объясните термины *алгоритмическое мышление* и *информационная культура*. Пользуясь каким-нибудь поисковым сервером, найдите в Интернете информацию об этих терминах.
- 5 **АНАЛИЗИРУЙТЕ!** Перечислите основные свойства алгоритмов. Проверьте, обладают ли программы, разработанные для управления исполнителями **Кенгуренок** и **Муравей**, этими свойствами.
- 6 Разработайте алгоритм для решения уравнений второй степени. Представьте этот алгоритм в виде блок-схемы.

# РЕАЛИЗАЦИЯ АЛГОРИТМОВ В СРЕДАХ ТЕКСТУАЛЬНОГО ПРОГРАММИРОВАНИЯ

### 3.1. Концепция действия

Согласно **концепции действия**, реализованной в языке ПАСКАЛЬ, компьютер является исполнителем, рабочая среда которого состоит из множества переменных и констант, объявленных в программе. В ходе выполнения программы исполнитель осуществляет над величинами из рабочей среды определенные действия (операции), например, сложение или вычитание, считывание с клавиатуры или вывод на экран и т.п. В результате этих действий значения переменных могут изменяться, в то время как значения констант – нет.

Действия, необходимые для обработки данных, и порядок их выполнения задаются с помощью операторов. Существуют две категории операторов:

- 1) простые операторы;
- 2) структурированные операторы.

**Простые операторы** не содержат других операторов. Простыми операторами являются:

- оператор присваивания;
- оператор вызова процедуры;
- оператор перехода;
- пустой оператор.

**Структурированные операторы** составлены из других операторов. Структурированными операторами являются:

- составной оператор **begin...end**;
- операторы условного перехода **if, case**;
- операторы цикла **for, while, repeat**;
- оператор **with**.

Операторы **if** и **case** применяются для программирования алгоритмов с разветвлениями, а операторы **for, while** и **repeat** – для программирования циклических алгоритмов. Напомним, что циклические алгоритмы используются для описания многократно повторяющихся действий, а алгоритмы с разветвлениями – для выбора необходимых действий в зависимости от условий из рабочей среды исполнителя.

Каждому оператору может предшествовать метка. Ссылка на метку указыва-

ется в операторе перехода. Напомним, что метка – это целое число без знака и что метка отделяется от оператора с помощью символа ":" (двоеточие).

В программах на языке ПАСКАЛЬ количество операторов в строке не ограничено – один оператор может занимать одну или более строк, а в одной строке может быть несколько операторов. В качестве разделителя операторов используются символ ";" (точка с запятой).

## 3.2. Выражения

Формулы для вычисления значений представляются на языке ПАСКАЛЬ в виде **выражений**. Выражения состоят из операндов (констант, переменных, ссылок на функции) и операций. Операции классифицируются следующим образом:

- мультипликативные операции: \*, /, **div**, **mod**, **and**;
- аддитивные операции: +, -, **or**;
- операции отношения: <, <=, =, >=, >, <>, **in**.

В языке ПАСКАЛЬ выражения записываются в строку, без использования индексов. Напомним, что индексы представляют собой символы, помещенные справа или слева (вверху или внизу) перед числом или буквой, для которых они указывают значение или смысл. Таким образом, математическое выражение  $\frac{a+b}{3}$  будет записано на языке ПАСКАЛЬ как (a+b) / 3, а математическое выражение  $x^2$  будет записано на языке ПАСКАЛЬ как x\*x.

Примеры:

	<u>Обычная форма записи</u>	<u>Запись на языке ПАСКАЛЬ</u>
1)	$\frac{a+b}{c+d}$	(a+b) / (c+d)
2)	$\frac{-b + \sin(b-c)}{2a}$	(-b+sin (b-c) ) / (2*a)
3)	$-\frac{1}{xy}$	-1 / (x*y)
4)	$p < q \& r > s$	(p<q) <b>and</b> (r>s)
5)	$\overline{x \vee y}$	<b>not</b> (x <b>or</b> y)
6)	$\frac{1}{a+b} > \frac{1}{c+d}$	1 / (a+b) > 1 / (c+d)

Вызов функции может появляться в выражении везде, где допустимо присутствие константы или переменной.

Примеры:

- abs (x) – абсолютное значение (|x|);
- sqr (x) – квадрат x ( $x^2$ );
- sqrt (x) – квадратный корень ( $\sqrt{x}$ ).

Список всех **предопределенных функций** языка ПАСКАЛЬ можно просмотреть с помощью системы поддержки среды программирования.

## Вопросы и упражнения

❶ Перепишите следующие выражения согласно правилам языка ПАСКАЛЬ:

a)  $a^2 + b^2;$

h)  $2\pi r;$

b)  $a^2 + 2ab + b^2;$

i)  $\pi r^2;$

c)  $(a + b)^2;$

j)  $x_1 x_2 \vee x_3 x_4;$

d)  $v_0 t + \frac{at^2}{2};$

k)  $\overline{x_1 \vee x_2};$

e)  $\frac{-b + \sqrt{b^2 - 4ac}}{2a};$

l)  $|x| < 3;$

f)  $\cos \alpha + \cos \beta;$

m)  $|z| < 6 \ \& \ |q| > 3,14;$

g)  $\cos (\alpha + \beta);$

n)  $x > 0 \ \& \ y > 8 \ \& \ R < 15.$

❷ Переведите выражения, записанные на языке ПАСКАЛЬ, в обычные:

a) `sqr (a) +sqr (b)`

e) `cos (ALFA-BETA)`

b) `2*a* (b+c)`

f) `sqr (a+b) / (a-b)`

c) `sqrt ( (a+b) / (a-b) )`

g) `x>0 or q<p`

d) `exp (x+y)`

h) `not (x and y)`

❸ Какие из следующих выражений на языке ПАСКАЛЬ содержат ошибки?

a) `(( (+x) ))`

h) `sinx+cosx`

b) `(( (x) ))`

i) `sqr (x)+sqr (y)`

c) `a<<b or c>d`

j) `sin (-x)`

d) `not q and p`

k) `sin-x`

e) `q and not p`

l) `cos (x+y)`

f) `not q and p`

m) `sin (abs (x) +abs (y) )`

g) `a+-b`

n) `sqrt (-y)`

### 3.3. Вычисление выражений

Под вычислением некоторого выражения понимается нахождение его значения. Результат вычисления зависит от значений операндов и от операции, применяемой к ним. Правила вычисления выражений – такие же, как и в математике:

- операции выполняются в соответствии с их приоритетом;
- в случае одинаковых приоритетов операции выполняются слева направо;
- первыми вычисляются значения выражений, заключенных в скобки.

Приоритеты операций приведены в *таблице 3.1*.

Таблица 3.1

Приоритеты операторов языка ПАСКАЛЬ

Категория	Операции	Приоритет
Унарные операции	<b>not</b> , @	Первый (наивысший)
Мультипликативные операции	<b>*</b> , <b>/</b> , <b>div</b> , <b>mod</b> , <b>and</b>	Второй
Аддитивные операции	<b>+</b> , <b>-</b> , <b>or</b>	Третий
Операции отношения	<b>&lt;</b> , <b>&lt;=</b> , <b>=</b> , <b>&gt;=</b> , <b>&gt;</b> , <b>&lt;&gt;</b> , <b>in</b>	Четвертый (наименьший)

*Пример:*

Пусть  $x = 2$  и  $y = 6$ . Тогда:

1)  $2 * x + y = 2 * 2 + 6 = 4 + 6 = 10;$

2)  $2 * (x + y) = 2 * (2 + 6) = 2 * 8 = 16;$

3)  $x + y / x - y = 2 + 6 / 2 - 6 = 2 + 3 - 6 = 5 - 6 = -1;$

4)  $(x + y) / x - y = (2 + 6) / 2 - 6 = 8 / 2 - 6 = 4 - 6 = -2;$

5)  $x + y / (x - y) = 2 + 6 / (2 - 6) = 2 + 6 / (-4) = 2 + (-1,5) = 0,5;$

6)  $x + y < 15 = 2 + 6 < 15 = 8 < 15 = \text{true};$

7)  $(x + y < 15) \text{ and } (x > 3) = (2 + 6 < 15) \text{ and } (2 > 3) = (8 < 15) \text{ and } (2 > 3) = \text{true and false} = \text{false}.$

Текущее значение выражения можно вывести на экран с помощью процедуры `writeln`:

```
writeln(<Выражение>)
```

Программа Р38 выводит на экран результаты вычисления выражений  $x * y + z$  и  $x + y < z - 1$ . 0. Текущие значения переменных  $x$ ,  $y$  и  $z$  считываются с клавиатуры.

```

Program P38;
{ Вычисление выражений }
var x, y, z : real;
begin
  writeln('Введите вещественные числа x, y, z:');
  readln(x, y, z);
  writeln(x*y+z);
  writeln(x+y<z-1.0);
end.

```

## Вопросы и упражнения

- ❶ Пусть  $x = 1$ ,  $y = 2$  и  $z = 3$ . Вычислите следующие выражения:

a)  $x+y+2*z$

f)  $x*(y+y)*z$

b)  $(x+y+2)*z$

g)  $(x*y+y)*z$

c)  $x*y+y*z$

h)  $x*(y+y)*z$

d)  $x*y<y*z$

i) **not** ( $x+y+z>0$ )

e)  $(x>y)$  **or**  $(6*x>y+z)$

j) **not** ( $x+y>0$ ) **and** **not** ( $z<0$ )

- ❷ Сформулируйте правила вычисления выражений на языке ПАСКАЛЬ.  
 ❸ Назовите приоритеты операций языка ПАСКАЛЬ.  
 ❹ Укажите порядок вычисления выражений.  
 ❺ Напишите программу, которая вычисляет выражения c) и g) из упражнения 1. Текущие значения вещественных переменных  $x$ ,  $y$  и  $z$  считываются с клавиатуры.

## 3.4. Тип выражений

Каждому выражению, в зависимости от множества его значений, ставится в соответствие определенный тип данных. Согласно концепции данных, реализованной в языке ПАСКАЛЬ, **тип выражения** определяется типами операндов и операциями, применяемыми к ним. Таким образом, тип выражения можно определить, не вычисляя его значения.

Типы результатов операций указаны в *таблице 3.2*.

Независимо от типа операция / (деление) возвращает результат типа **real**, а „операции отношения” возвращают результат типа **boolean**.

Типы результатов операций

Операция	Тип операндов	Тип результата
+, -, *	integer	integer
	Один - integer, другой - real	real
/	integer или real	real
div	integer	integer
mod	integer	integer
not, and, or	boolean	boolean
<, <=, =, >=, >, <>	Идентичные типы	boolean
	Совместимые типы	boolean
	Один - integer, другой - real	boolean

Для того чтобы определить тип выражения, факторы, термы и простые выражения рассматриваются в порядке их вычисления, причем тип каждой составной части определяется с помощью *таблицы 3.2*.

Например, рассмотрим выражение:

```
(x>i) or (6*i<sin(x/y))
```

в котором *i* - переменная типа *integer*, а *x* и *y* - переменные типа *real*.

Определим тип составных частей и всего выражения в порядке выполнения вычислений:

1)	<code>x&gt;i</code>	<code>boolean;</code>
2)	<code>6*i</code>	<code>integer;</code>
3)	<code>x/y</code>	<code>real;</code>
4)	<code>sin(x/y)</code>	<code>real;</code>
5)	<code>6*i &lt; sin(x/y)</code>	<code>boolean;</code>
6)	<code>(x&gt;i) or (6*i&lt;sin(x/y))</code>	<code>boolean.</code>

Таким образом, рассматриваемое выражение относится к типу *boolean*.

Так как в процессе определения типов конкретные значения соответствующих выражений не вычисляются, интервальные типы расширяются до базовых.

Например, при следующих описаниях:

```
type T1=1..10; { интервал типа integer }
      T2=11..20; { интервал типа integer }
var i:T1;
    j:T2;
```



имеем:

	<u>Выражение</u>	<u>Тип выражения</u>
1)	<code>i+j</code>	<code>integer;</code>
2)	<code>i mod j</code>	<code>integer;</code>
3)	<code>i/j</code>	<code>real;</code>
4)	<code>sin(i+j)</code>	<code>real;</code>
5)	<code>i&gt;j</code>	<code>boolean.</code>

и т.д.

В зависимости от типа выражения разделяются на:

- арифметические (`integer` или `real`);
- порядковые (`integer`, `boolean`, `char`, перечисляемые);
- логические (`boolean`).

Как правило, арифметические выражения используются в вычислениях (оператор присваивания), порядковые выражения – в операторах **case** и **for**, а логические – в операторах **if**, **repeat** и **while**.

## Вопросы и упражнения

- ❶ Как определяется тип выражения в языке ПАСКАЛЬ?
- ❷ При наличии объявлений:

```
var x, y : real;  
    i, j : integer;  
    p, q : boolean;  
    r : char;  
    s : (A, B, C, D, E, F, G, H);
```

определите типы следующих выражений:

a) <code>i mod 3</code>	i) <code>sqr(i)-sqr(j)</code>
b) <code>i/3</code>	j) <code>sqr(x)-sqr(y)</code>
c) <code>i mod 3 &gt; j div 4</code>	k) <code>trunc(x)+trunc(y)</code>
d) <code>x+y/(x-y)</code>	l) <code>chr(i)</code>
e) <code>not(x&lt;i)</code>	m) <code>ord(r)</code>
f) <code>sin(abs(i)+abs(j))</code>	n) <code>ord(s)&gt;ord(r)</code>
g) <code>sin(abs(x)+abs(y))</code>	o) <code>pred(E)</code>
h) <code>p and (cos(x)&lt;=sin(y))</code>	p) <code>(-x+sin(x-y))/(2*i)</code>

- ❸ Тип выражения можно узнать из текстовой формы результатов, выведенных на экран с помощью оператора `writeln (<Выражение>)`.

Примеры:

	<u>Результат, выведенный на экран</u>	<u>Тип выражения</u>
1)	100	integer;
2)	1.000000000000E+02	real;
3)	true	boolean.

Напишите соответствующие программы и на основании текстовой формы результатов, выведенных на экран, определите типы следующих выражений:

a)	1+1.0	f)	<b>not</b> (x>y)
b)	1/1+1	g)	pred (9)>succ (7)
c)	9*3 <b>mod</b> 4	h)	15 <b>div</b> ord(3)
d)	4*x>9*y	i)	trunc (x)+round (6*y)
e)	chr (65)	j)	sqr (3)-sqrt (16)

где переменные x и y являются переменными типа real (вещественный).

4 Даны описания:

```

type T1=1..10;
      T2=11..20;
      T3='A'..'Z';
      T4=(A, B, C, D, E, F, G, H);
var  i : T1;
      j : T2;
      k : T3;
      m : 'C'..'G';
      n : T4;
      p : C..G;
      q : boolean;

```

Найдите типы следующих выражений:

a)	i-j	j)	ord (m)
b)	i <b>div</b> j	k)	n>p
c)	6.3*i	l)	ord (n)
d)	cos (3*i-6*j)	m)	succ (n)
e)	4*i>5*j	n)	pred (p)
f)	k<m	o)	ord (p)
g)	k<>m	p)	ord (k)>ord (m)
h)	chr (i)	q)	(i>j) <b>and</b> q
i)	ord (k)	r)	<b>not</b> (i+j>0) <b>or</b> q

### 3.5. Оператор присваивания

Оператор присваивания имеет вид:

*<Переменная> := <Выражение>*

При выполнении оператора присваивания происходит следующее:

- а) вычисляется выражение, стоящее в правой части;
- б) полученное значение присваивается переменной, стоящей в левой части.

Примеры:

- |                                  |   |
|----------------------------------|---|
| 1) <code>x:=1</code>             | 4) <code>p:=not q</code>                |
| 2) <code>y:=x+3</code>           | 5) <code>q:=(a&lt;b) or (x&lt;y)</code> |
| 3) <code>z:=sin(x)+cos(y)</code> | 6) <code>c:='A'</code>                  |

Отметим, что символ “:=” (читается «присвоить») означает присваивание и не следует путать его с операцией отношения “=” (равно).

Присваивание возможно только тогда, когда переменная и результат вычисления выражения **совместимы с точки зрения присваивания**. В противном случае возникает ошибка.

Переменная и результат вычисления выражения являются совместимыми с точки зрения присваивания, если справедливо одно из следующих утверждений:

- 1) тип переменной и тип результата идентичны;
- 2) тип результата является интервалом типа переменной;
- 3) оба типа являются интервалами одного и того же типа, а тип результата принадлежит интервальному типу переменной;
- 4) тип переменной – real, а тип результата – integer или его интервал.

В качестве примера рассмотрим следующую программу:

```
Program P39;  
  { Совместимость с точки зрения присваивания }  
type T1=1..10; { интервал типа integer }  
      T2=5..15; { интервал типа integer }  
var i : T1;  
    j : T2;  
    k, m, n : integer;  
    x : real;  
begin  
  write('k=');  
  readln(k);  
  i:=k; { верно для 1<=k<=10 }  
  write('m=');  
  readln(m);  
  j:=m; { верно для 5<=m<=15 }  
  write('n=');  
  readln(n);
```

```

i:=n+5; { верно для -4<=n<=5 }
j:=n+2; { верно для 3<=n<=13 }
x:=i+j;
writeln('x=', x);
end.

```

Программа P39 будет работать без ошибок только при следующих значениях на входе:

$$1 \leq k \leq 10; 5 \leq m \leq 15; 3 \leq n \leq 5.$$

Очевидно, что в программе P39 операторы вида

`k:=x`

`i:=x+1`

`j:=sin(i)`

являются ошибочными, так как тип выражений `x`, `x+1`, `sin(i)` – `real`, а тип переменных `k`, `i`, `j` – `integer` или его интервал.

## Вопросы и упражнения

- ❶ Как выполняется оператор присваивания?
- ❷ Объясните термин «совместимость с точки зрения присваивания».
- ❸ Даны следующие описания:

```

type Zi = (L, Ma, Mi, J, V, S, D);
      Culoare = (Galben, Verde, Albastru, Violet);
var i, j, k : integer;
    z : Zi;
    c : Culoare;
    x : real;

```

Какие из следующих выражений являются правильными?

a) `i:=12`

f) `c:=Verde`

b) `j:=ord(i)`

g) `z:=D`

c) `x:=ord(z)+1`

h) `c:=Pred(Galben)`

d) `k:=ord(x)+2`

i) `x:=Succ(z)`

e) `c:=i+4`

j) `i:=Succ(c)`

- ❹ При каких значениях переменной `j` программа P40 будет работать без ошибок?

```

Program P40;
var i : -10..+10;
    j : integer;
begin
  write('j=');
  readln(j);
  i:=j+15;
  writeln('i=', i);
end.

```

### 3.6. Команда вызова процедуры

**Процедура** представляет собой подпрограмму, к которой в процессе выполнения программы можно обращаться произвольное число раз. Каждая процедура имеет свое имя, например: `readln`, `writeln`, `CitireDate`, `A15` и т. д. Язык ПАСКАЛЬ содержит ряд стандартных процедур, известных любой программе: `read`, `readln`, `write`, `writeln`, `get`, `put`, `new` и т. д. В дополнение к ним программист может создавать собственные процедуры.

Оператор *вызова процедуры* запускает на выполнение процедуру с соответствующим именем. Данный оператор имеет вид:

*Имя процедуры (Фактический параметр\_1, Фактический параметр\_2, ...)*

Если у процедуры нет параметров, то оператор вызова состоит только из ее имени.

Примеры:

1) `readln(x)`

4) `Exit`

2) `readln(x, y, z)`

5) `writeln(x+y, sin(x))`

3) `CitireDate(f, t)`

6) `writeln(2*x)`

Тип каждого фактического параметра и порядок его появления в списке указываются в разделе описаний соответствующих процедур. Правила составления списка фактических параметров будут изучены в следующих классах.

#### Вопросы и упражнения

❶ Для чего необходим оператор вызова процедуры?

❷ Даны следующие операторы:

a) `readln(x, y, z, q)`

b) `CitireDate(ff, tt)`

c) `Halt`

d) `writeln('x=', x, 'y=', y)`

e) `writeln('x+y=', x+y, 'sin(x)=' , sin(x))`

Укажите имена вызываемых процедур, число фактических параметров каждой процедуры, а также назовите эти фактические параметры.

❸ Напишите на языке ПАСКАЛЬ программу, которая считывает с клавиатуры длины сторон треугольника и выводит на экран его площадь.

### 3.7. Вывод алфавитно-цифровой информации на экран

В большинстве версий языка ПАСКАЛЬ экран монитора является стандартным устройством вывода. Как правило, экран разделяется на условные зоны, называемые символьными. Указанные зоны образуют 25 строк по 80 символов в каждой. Зона, в которой будет выведен текущий символ, обозначается курсором.

Данные выводятся на экран с помощью процедуры `write(x)` или `writeln(x)`.

Оператор вызова процедуры

```
write(x1, x2, ..., xn)
```

эквивалентен последовательности:

```
write(x1); write(x2); ...; write(xn).
```

Фактические параметры процедур `write` или `writeln` называются **параметрами вывода**. Они могут иметь следующий вид:

*e*

*e:w*

*e:w:f*

где *e* – выражение типа `integer`, `real`, `boolean`, `char` или строкового типа, значение которого нужно вывести на экран; *w* и *f* являются выражениями типа `integer` и называются **спецификаторами формата**. Значение выражения *w* указывает минимальное число символов, используемых для вывода значения *e*; если для представления значения *e* требуется меньше, чем *w* символов, то ему предшествует такое число пробелов, чтобы было записано точно *w* символов (рис. 3.1).







<code>write (-1234)</code>	
<code>write (-1234:10)</code>	
<code>write ('a')</code>	
<code>write ('a':5)</code>	
<code>write ('a', 'b')</code>	
<code>write ('a':5, 'b':5)</code>	

Рис. 3.1. Значение спецификатора формата *w*

Спецификатор формата *f* может присутствовать только в том случае, когда *e* является выражением типа `real`. Данный параметр указывает количество цифр после запятой в представлении значения выражения *e* с фиксированной точкой, без масштабного множителя. При отсутствии *f* значение *e* записывается с плавающей точкой, с масштабным множителем (рис. 3.2).

<code>write (-1234.567890)</code>	<code>- 1 . 2 3 4 5 6 7 8 9 0 0 E + 0 3</code>
<code>write (-1234.567890:20)</code>	<code>- 1 . 2 3 4 5 6 7 8 9 0 0 E + 0 3</code>
<code>write (-1234.567890:20:1)</code>	<code>- 1 2 3 4 . 6</code>
<code>write (-1234.567890:20:4)</code>	<code>- 1 2 3 4 . 5 6 7 9</code>

Рис. 3.2. Значение спецификатора формата *f*

Разница между процедурами `write` и `writeln` состоит в том, что после вывода данных `write` оставляет курсор в текущей строке, тогда как `writeln` переводит курсор на начало следующей строки. Рациональное использование процедур `write`, `writeln` и параметров размера поля обеспечивает вывод данных в форме, удобной для чтения. При выводе на экран нескольких значений рекомендуется указывать соответствующие им идентификаторы или сопровождать их комментариями.

Примеры:

- 1) `write('Сумма введенных чисел')`
- 2) `writeln(s:20)`
- 3) `writeln('Сумма=', s)`
- 4) `writeln('s=', s)`
- 5) `writeln('x=', x, 'y':5, y, 'z':5, z)`

## Вопросы и упражнения

- ❶ Для чего нужен спецификатор формата?
- ❷ Как называются фактические параметры процедур `write` и `writeln`?
- ❸ Определите форматы данных, выводимых на экран следующими программами:

```

Program P41;
{ Вывод данных типа integer }
var i : integer;
begin
  i:=-1234;
  writeln(i);
  writeln(i:1);
  writeln(i:8);
  writeln(i, i);
  writeln(i:8, i:8);
  writeln(i, i, i);

```



```
writeln(i:8, i:8, i:8);
end.
```

```
Program P42;
{ Вывод данных типа real }
var x : real;
begin
  x:=-1234.567890;
  writeln(x);
  writeln(x:20);
  writeln(x:20:1);
  writeln(x:20:2);
  writeln(x:20:4);
  writeln(x, x, x);
  writeln(x:20, x:20, x:20);
  writeln(x:20:4, x:20:4, x:20:4);
end.
```

```
Program P43;
{ Вывод данных типа boolean }
var p : boolean;
begin
  p:=false;
  writeln(p);
  writeln(p:10);
  writeln(p, p);
  writeln(p:10, p:10);
end.
```

```
Program P44;
{ Вывод символьных строк }
begin
  writeln('abc');
  writeln('abc':10);
  writeln('abc', 'abc');
  writeln('abc':10, 'abc':10);
end.
```

- ④ Напишите программу, которая выводит на экран значения 1234567890, 123, 123.0 и true следующим образом:

```
1234567890
123
123.0
true
1234567890
123
123.000
true
```

## 3.8. Ввод данных с клавиатуры

Как правило, стандартным устройством ввода является клавиатура. Ввод данных с клавиатуры выполняется с помощью стандартных процедур `read` или `readln`. Список фактических параметров процедуры `read` или `readln` может включать переменные типа `integer`, `real`, `char` и строкового типа.

Оператор вызова процедуры

```
read(x)
```

выполняет следующие действия: если переменная `x` является переменной типа `integer` или `real`, тогда считывается вся строка символов, представляющая целое или вещественное значение; если переменная `x` является переменной типа `char`, то процедура считывает только один символ.

Оператор вызова процедуры

```
read(x1, x2, ..., xn)
```

эквивалентен последовательности:

```
read(x1); read(x2); ...; read(xn) .
```

Числа, вводимые с клавиатуры, должны разделяться пробелами или символами конца строки. Пробелы, стоящие перед самым числом, игнорируются. Строка символов, представляющая собой число, должна соответствовать синтаксису числовых констант соответствующего типа. В противном случае возникает ошибка ввода-вывода.

Например, рассмотрим программу:

```
Program P45;
{ Считывание чисел с клавиатуры }
var i, j : integer;
    x, y : real;
begin
    read(i, j, x, y);
    writeln('Были введены:');
    writeln('i=', i);
    writeln('j=', j);
    writeln('x=', x);
    writeln('y=', y);
end.
```

в которой считываются с клавиатуры значения переменных `i`, `j`, `x`, `y`. После запуска программы на выполнение пользователь набирает:

```
1<ENTER>
2<ENTER>
3.0<ENTER>
4.0<ENTER>
```

На экран будет выведено:

```
Были введены:  
i=1  
j=2  
x=3.0000000000E+00  
y=4.0000000000E+00
```

Если данные вводить одной строкой, то результат не изменится:

```
1 2 3.0 4.0<ENTER>
```

В случае необходимости целые числа, введенные пользователем, переводятся в вещественные значения.

Например, в программе P45 пользователь может набрать на клавиатуре

```
1 2 3 4<ENTER>
```

Процедура `readln` считывает данные точно так же, как и процедура `read`. Однако после считывания последнего значения оставшиеся символы текущей строки игнорируются. В качестве примера рассмотрим программу P46:

```
Program P46;  
{Использование процедуры readln}  
var i, j : integer;  
      x, y : real;  
begin  
  writeln('Использование процедуры read');  
  read(i, j);  
  read(x, y);  
  writeln('Были введены:');  
  writeln('i=', i, ' j=', j, ' x=', x, ' y=', y);  
  writeln('Использование процедуры readln');  
  readln(i, j);  
  readln(x, y);  
  writeln('Были введены:');  
  writeln('i=', i, ' j=', j, ' x=', x, ' y=', y);  
end.
```

При выполнении операторов

```
read(i, j);  
read(x, y);
```

числовые значения из строки

```
1 2 3 4<ENTER>
```

введенной пользователем, будут присвоены соответственно переменным `i`, `j`, `x`, `y`. При выполнении оператора

```
readln(i, j)
```

числовые значения 1 и 2 из строки

```
1 2 3 4<ENTER>
```

будут присвоены переменным *i* и *j*. Числа 3 и 4 игнорируются. Затем компьютер выполняет команду

```
readln(x, y)
```

т. е. ожидается набор значений для *x* и *y*. Отметим, что при вызове процедуры `readln` без параметров компьютер ожидает нажатия клавиши <ENTER>. Такой вызов процедуры используется с целью приостановки выполнения программы, предоставляя тем самым пользователю возможность прочесть результаты, выведенные до этого на экран.

Для того чтобы сообщить пользователю, какие данные нужно вводить, рекомендуется перед вызовом процедур `read (...)` и `readln (...)` выводить на экран соответствующие информирующие сообщения.

Примеры:

- 1) `write('Введите два числа:'); readln(x, y);`
- 2) `write('Введите целое число:'); readln(i);`
- 3) `write('x='); readln(x);`
- 4) `write('Ответьте да/D или нет/N:'); readln(c);`

## Вопросы и упражнения

- ❶ Как разделяются числовые данные, вводимые с клавиатуры?
- ❷ Каковы различия между процедурами `read` и `readln`?
- ❸ Дана следующая программа:

```
Program P47;  
var i : integer;  
    c : char;  
    x : real;  
begin  
    readln(i);  
    readln(c);  
    readln(x);  
    writeln('i=', i);  
    writeln('c=', c);  
    writeln('x=', x);  
    readln;  
end.
```

Определите результаты, которые будут выведены на экран при вводе следующих данных:

a) 

1
2
3

b) 

1	2	3
5	6	7
8	9	0

c) 

123
456
789

d) 

123	456	789
abc	def	ghi
890	abc	def

### 3.9. Пустой оператор

Выполнение пустого оператора никак не влияет на значения переменных, используемых в программе. Так как операторы программы разделяются с помощью “;”, присутствие пустого оператора отмечается появлением этого символа.

Например, в тексте:

```
x:=4;;; y:=x+1
```

есть 5 операторов, 3 из которых – пустые.

Обычно пустой оператор используется на этапах разработки и отладки сложных программ. Хотя пустой оператор не выполняет никаких действий, его (точнее, символа ;) присутствие или отсутствие может повлиять на ход программы.

### 3.10. Оператор **if**

Оператор выбора **if** выполняет одно из двух возможных действий в зависимости от значения некоторого условия – логического выражения. Простая форма данного оператора:

**if** *Логическое выражение* **then** *Оператор*,

а полная форма включает и альтернативу **else**:

**if** *Логическое выражение* **then** *Оператор\_1* **else** *Оператор\_2*

Логическое выражение, входящее в состав оператора **if**, называется **условием**.

Выполнение оператора **if** начинается с проверки условия. Если результатом проверки является **true**, то выполняется оператор, стоящий после ключевого слова **then**. Если условие принимает значение **false**, то выполняется оператор, стоящий после ключевого слова **else** (если оно есть), или управление передается команде, следующей непосредственно за оператором **if**.

В следующей программе оператор **if** используется для определения максимального из двух чисел *x* и *y*, считываемых с клавиатуры.

```
Program P48;  
{ Определение максимального из двух чисел }  
var x, y, max : real;  
begin  
  writeln('Введите два числа:');  
  write('x='); readln(x);  
  write('y='); readln(y);  
  if x>=y then max:=x else max:=y;  
  writeln('max=', max);  
  readln;  
end.
```

Следующая программа переводит римские цифры: I (один), V (пять), X (десять), L (пятьдесят), C (сто), D (пятьсот), M (тысяча), считываемые с клавиатуры, в соответствующие им числа в десятичной системе счисления.

```

Program P49;
  { Перевод римских цифр }
var i : integer; c : char;
begin
  i:=0;
  writeln('Введите одну из римских цифр');
  writeln('romane I, V, X, L, C, D, M');
  readln(c);
  if c='I' then i:=1;
  if c='V' then i:=5;
  if c='X' then i:=10;
  if c='L' then i:=50;
  if c='C' then i:=100;
  if c='D' then i:=500;
  if c='M' then i:=1000;
  if i=0 then writeln(c, ' - не является римской цифрой')
    else writeln(i);
  readln;
end.

```

Отметим, что в языке ПАСКАЛЬ символ ";" не является частью оператора, а используется в качестве разделителя. Следовательно, если в команде:

```
if B then S
```

перед S поставить пустой оператор

```
if B then; S
```

тогда S не будет входить в состав оператора **if**. В таком случае S будет выполняться независимо от значения B.

Если в операторе

```
if B then I else J
```

после I поставить символ ";", то получим синтаксически неправильно составленную программу:

```
if B then I; else J
```

В данном случае текст **else** J интерпретируется как оператор, стоящий после оператора **if**.

## Вопросы и упражнения

- 1 Для чего необходим оператор **if**?
- 2 Какие значения будет принимать переменная x после выполнения каждого из следующих операторов? Подразумевается, что a=18, b=-15 и p=true.

a) **if** a>b **then** x:=1 **else** x:=4;

- b) **if** a<b **then** x:=15 **else** x:=-21;
- c) **if** p **then** x:=32 **else** x:=638;
- d) **if not** p **then** x:=0 **else** x:=1;
- e) **if** (a<b) **and** p **then** x:=-1 **else** x:=1;
- f) **if** (a>b) **or** p **then** x:=-6 **else** x:=-5;
- g) **if not** (a>b) **then** x:=19 **else** x:=-2;
- h) **if** (a=b) **or** p **then** x:=89 **else** x:=-15.

③ Напишите программу, которая вычисляет значение одной из следующих функций:

a) 
$$y = \begin{cases} 2x, & x \geq 0; \\ \frac{x}{2}, & x < 0; \end{cases}$$

b) 
$$y = \begin{cases} x+3, & x > 5; \\ x-3, & x \leq 5; \end{cases}$$

c) 
$$y = \begin{cases} x, & x \geq 3; \\ x+4, & x < 3; \end{cases}$$

d) 
$$y = \begin{cases} x, & |x| > 5; \\ 2x, & |x| \leq 5. \end{cases}$$

Например, для  $y = \begin{cases} x+6, & x > 4; \\ x-3, & x \leq 4; \end{cases}$

получаем:

```
Program P50;
var x, y : real;
begin
  write('x='); readln(x);
  if x>4 then y:=x+6 else y:=x-3;
  writeln('y=', y);
  readln;
end.
```

④ Какие результаты выведет на экран следующая программа?

```
Program P51;
var x, y : real;
begin
  write('x='); readln(x);
  y:=x;
  if x>0 then; y:=2*x;
  writeln('y=', y);
  readln;
end.
```

⑤ Прокомментируйте сообщения, выводимые на экран в процессе компиляции программы P52:

```
Program P52;
{ Ошибка }
```



```

var x, y : real;
begin
  write('x=');
  readln(x);
  if x>4 then y:=x+6;
           else y:=x-3;
  writeln('y=', y); readln;
end.

```

- ⑥ Напишите программу, которая переводит десятичные числа 1, 5, 10, 50, 100, 500 и 1000, считываемые с клавиатуры, в римские.

### 3.11. Оператор case

Оператор множественного выбора **case** состоит из выражения, называемого **переключателем (селектором)**, списка констант и соответствующего ему списка команд.

Данный оператор имеет вид:

```

case Выражение of
  Константа_1: Оператор;
  Константа_2: Оператор;
  ...
end

```

Селектор должен относиться к порядковому типу. Константы выбора должны быть совместимыми с типом селектора и не могут повторяться.

Пример:

```

var i : integer; c : char; a, b, y : real;

```

```

1) case i of
    0, 2, 4, 6, 8 : writeln('Четная цифра');
    1, 3, 5, 7, 9 : writeln('Нечетная цифра');
end

```

```

2) case c of
    '+' : y:=a+b;
    '-' : y:=a-b;
    '*' : y:=a*b;
    '/' : y:=a/b;
end

```

Выполнение оператора **case** начинается с вычисления значения селектора. Если селектор принимает одно из значений констант выбора, то выполняется оператор, соответствующий этой константе.

В следующей программе оператор **case** используется для перевода римских цифр в десятичные числа.

```

Program P53;
{ Перевод римских цифр в десятичные }
var i : integer; c : char;
begin
  i:=0;
  writeln('Введите одну из римских цифр');
  writeln('I, V, X, L, C, D, M');
  readln(c);
  case c of
    'I' : i:=1;
    'V' : i:=5;
    'X' : i:=10;
    'L' : i:=50;
    'C' : i:=100;
    'D' : i:=500;
    'M' : i:=1000;
  end;
  if i=0 then writeln(c, ' - не является римской цифрой')
    else writeln(i);
  readln;
end.

```

Отметим, что в некоторых версиях языка синтаксис и семантика команды **case** были изменены. Список вариантов может включать команду, которой предшествует ключевое слово **else** (в некоторых версиях **otherwise**). Константы выбора можно заменить интервалами вида

*<Константа> .. <Константа>*

*Пример (Turbo PASCAL 7.0, Free PASCAL):*

```

Program P54;
{ Модель карманного калькулятора }
var a, b : real;
    c : char;
begin
  write('a='); readln(a);
  write('b='); readln(b);
  write('Код операции'); readln(c);
  case c of
    '+' : writeln('a+b=', a+b);
    '-' : writeln('a-b=', a-b);
    '*' : writeln('a*b=', a*b);
    '/' : writeln('a/b=', a/b);
  else writeln('Недопустимый код операции');
  end;
  readln;
end.

```

## Вопросы и упражнения

- ❶ Как выполняется оператор **case**? Каким должен быть тип селектора?
- ❷ Какие константы можно использовать в качестве констант выбора?
- ❸ Замените оператор **case** программы P54 последовательностью эквивалентных ему операторов **if**.
- ❹ Используя оператор **case**, напишите программу, которая переводит десятичные числа 1, 5, 10, 50, 100, 500, 1000, считываемые с клавиатуры, в римские.
- ❺ Что появится на экране в процессе выполнения программы P55?

```
Program P55;  
type Semnal=(Rosu, Galben, Verde);  
var s : Semnal;  
begin  
  s:=Verde;  
  s:=pred(s);  
  case s of  
    Rosu : writeln('СТОП');  
    Galben : writeln('ВНИМАНИЕ');  
    Verde : writeln('СТАРТ');  
  end;  
  readln;  
end.
```

- ❻ Прокомментируйте следующие программы:

```
Program P56;  
{ Ошибка }  
var x : real;  
begin  
  writeln('x='); readln(x);  
  case x of  
    0,2,4,6,8 : writeln('Четная цифра');  
    1,3,5,7,9 : writeln('Нечетная цифра');  
  end;  
  readln;  
end.
```

```
Program P57;  
{ Ошибка }  
var i : 1..4;  
begin  
  write('i='); readln(i);  
  case i of  
    1 : writeln('Один');  
    2 : writeln('Два');  
    3 : writeln('Три');  
    4 : writeln('Четыре');  
    5 : writeln('Пять');  
  end;  
  readln;  
end.
```

## 3.12. Оператор **for**

Оператор **for** предназначен для повторного выполнения другого оператора в зависимости от значения управляющей переменной. Данный оператор имеет вид:

```
for Переменная:= Выражение_1 to Выражение_2 do Оператор  
for Переменная:= Выражение_1 downto Выражение_2 do Оператор
```

Переменная, находящаяся после ключевого слова **for**, называется управляющей переменной или параметром цикла. Эта переменная должна принадлежать некоторому порядковому типу.

Значения выражений, входящих в состав оператора **for**, должны быть совместимыми с точки зрения присваивания, с параметром цикла. Эти выражения проверяются один раз в начале цикла. Первое выражение указывает исходное значение параметра цикла, а второе – конечное значение.

Оператор, стоящий после ключевого слова **do**, выполняется для каждого значения из диапазона, определяемого начальным и конечным значениями.

Если в команде **for** используется ключевое слово **to**, то значение параметра цикла увеличивается при каждом повторении, переходя к значению, следующему за текущим. Если исходное значение больше конечного, оператор, расположенный после ключевого слова **do**, не выполняется ни разу. Если в операторе **for** используется ключевое слово **downto**, то значение параметра цикла уменьшается при каждом повторении, переходя к значению, предшествующему текущему. Если начальное значение меньше конечного, то оператор, расположенный после ключевого слова **do**, не выполняется ни разу.

*Пример:*

```
Program P58;  
  { Оператор for }  
var i : integer;  
      c : char;  
begin  
  for i:=0 to 9 do write(i:2);  
  writeln;  
  for i:=9 downto 0 do write(i:2);  
  writeln;  
  for c:='A' to 'Z' do write(c:2);  
  writeln;  
  for c:='Z' downto 'A' do write(c:2);  
  writeln;  
  readln;  
end.
```

Результаты, выводимые на экран:

```
0 1 2 3 4 5 6 7 8 9  
9 8 7 6 5 4 3 2 1 0
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Значения управляющей переменной не могут быть изменены внутри цикла, т. е.

- 1) управляющей переменной не присваиваются никакие значения;
- 2) текущая управляющая переменная не может быть управляющей переменной другой вложенной команды **for**;
- 3) нельзя вызывать процедуры `read` и `readln`, в которых указывается управляющая переменная.

После выхода из команды **for** значение управляющей переменной не определено, за исключением случая, когда выход из цикла осуществляется принудительно, с помощью оператора безусловного перехода **goto**.

Оператор **for** используется для программирования итеративных алгоритмов, в которых число повторений известно. В качестве примера рассмотрим программы P59, P60 и P61, которые вычисляют соответственно  $n!$ ,  $x^n$  и сумму

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n}.$$

```
Program P59;
{ Вычисление факториала }
var n, i, f : 0..MaxInt;
begin   write('n='); readln(n);
        f:=1;
        for i:=1 to n do f:=f*i;
        writeln('n!=', f);
        readln;
end.
```

```
Program P60;
{ Вычисление x в степени n }
var x, y : real;
    n, i : 0..MaxInt;
begin
    write('x='); readln(x);
    write('n='); readln(n);
    y:=1;
    for i:=1 to n do y:=y*x;
    writeln('y=', y);
    readln;
end.
```

```
Program P61;
{ Вычисление суммы 1 + 1/2 + 1/3 + ... + 1/n }
var n, i : 1..MaxInt;
    s : real;
```

```

begin
  write('n=');
  readln(n);
  s:=0;
  for i:=1 to n do s:=s+1/i;
  writeln('s=', s);
  readln;
end.

```

## Вопросы и упражнения

- ❶ Как выполняется оператор **for**?
- ❷ Что выведет на экран программа P62?

```

Program P62;
type Zi = (L, Ma, Mi, J, V, S, D);
var z : Zi;
begin
  for z:=L to S do writeln(ord(z));
  readln;
  for z:=D downto Ma do writeln(ord(z));
  readln;
end.

```

- ❸ Даны объявления:

```

var i, j, n : integer;
    x, y : real;
    c : char;

```

Какие из следующих команд являются синтаксически правильными?

- a) **for** i:=-5 **to** 5 **do** j:=i+3;
- b) **for** i:=-5 **to** 5 **do** i:=j+3;
- c) **for** j:=-5 **to** 5 **do** i:=j+3;
- d) **for** i:=1 **to** n **do** y:=y/i;
- e) **for** x:=1 **to** n **do** y:=y/x;
- f) **for** c:='A' **to** 'Z' **do** writeln(ord(c));
- g) **for** c:='Z' **downto** 'A' **do** writeln(ord(c));
- h) **for** i:=-5 **downto** -10 **do** readln(i);
- i) **for** i:=ord('A') **to** ord('A')+ 9 **do** writeln(i);
- j) **for** c:='0' **to** '9' **do** writeln(c, ord(c):3);
- k) **for** j:=i/2 **to** i/2+10 **do** writeln(j).

④ Даны объявления

```
var i, m, n : integer;
```

Сколько раз будут выполнены процедуры `writeln(i)` и `writeln(2*i)`, входящие в состав операторов

```
for i:=m to n do writeln(i);  
for i:=m to n do writeln(2*i);
```

если:

a) `m=1, n=5;`

c) `m=3, n=3;`

b) `m=3, n=5;`

d) `m=5, n=3?`

⑤ Напишите программу, которая выводит на экран коды символов 'A', 'B', 'C', ..., 'Z'.

⑥ Вычислите для первых  $n$  элементов:

a) `1 + 3 + 5 + 7 + ...` и `1 * 3 * 5 * 7 * ...;`

b) `2 + 4 + 6 + 8 + ...` и `2 * 4 * 6 * 8 * ...;`

c) `3 + 6 + 9 + 12 + ...` и `3 * 6 * 9 * 12 * ...;`

d) `4 + 8 + 12 + 16 + ...` и `4 * 8 * 12 * 16 * ...`

Например: При  $n=3$  имеем  $1 + 3 + 5 = 9$ ;  $1 * 3 * 5 = 15$ .

⑦ Вычислите сумму первых  $n$  элементов:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \dots$$

Указание. Используйте внутри цикла команду `if odd(...) then ... else ...`.

## 3.13. Составной оператор

Составной оператор имеет вид:

```
begin
```

```
    Оператор_1;
```

```
    Оператор_2;
```

```
    ...
```

```
end
```

Примеры:

```
1) begin  
    a:=x+12;  
    p:=q and r;  
    writeln(p)  
end
```



```
2) begin
    write('x=');
    readln(x)
end
```

Ключевые слова **begin** и **end** выступают в роли “скобок”. Последовательность операторов, заключенных в данные скобки, является, с точки зрения языка, одним оператором. Таким образом, составной оператор используется для того, чтобы несколько операторов поместить в те места программы, где разрешается наличие только одного оператора (см. операторы **if**, **case**, **for** и др.).

*Примеры:*

```
1) if a>0 then begin x:=a+b; y:=a*b end
    else begin x:=a-b; y:=a/b end;
```

```
2) case c of
    '+' : begin y:=a+b; writeln('Adunarea') end;
    '-' : begin y:=a-b; writeln('Scaderea') end;
    '*' : begin y:=a*b; writeln('Inmultirea') end;
    '/' : begin y:=a/b; writeln('Impartirea') end;
end;
```

```
3) for i:=1 to n do
    begin
        write('x=');
        readln(x);
        s:=s+x
    end;
```

Отметим, что тело любой программы является составным оператором, так как оно представляет собой последовательность команд, заключенных в “скобки” **begin** и **end**.

Так как символ “;” не заканчивает, а разделяет операторы, то присутствие его перед ключевым словом **end** не обязательно. Однако многие программисты вставляют данный символ с целью продолжить, в случае необходимости, список операторов. Напоминаем, что дополнительное появление символа “;” означает вставку пустого оператора.

Для того чтобы программы были удобными для чтения, слова **begin** и **end** пишутся строго одно под другим, а операторы внутри “скобок” смещаются на несколько позиций вправо. Если составной оператор **begin...end** включается в состав других команд (**if**, **case**, **for** и др.), то ключевые слова **begin** и **end** смещаются вправо.

В качестве примера рассмотрим программу P63, в которой вычисляется среднее арифметическое  $n$  чисел, считываемых с клавиатуры.

```
Program P63;
{ Среднее арифметическое n чисел }
var x, suma, media : real;
    i, n : integer;
```

```

begin
  write('n='); readln(n);
  suma:=0;
  writeln('Введите ', n, ' чисел:');
  for i:=1 to n do
    begin
      write('x='); readln(x);
      suma:=suma+x;
    end;
  if n>0 then
    begin
      media:=suma/n;
      writeln('среднее=', media);
    end
  else writeln('среднее=*****');
  readln;
end.

```

## Вопросы и упражнения

- ❶ Для чего необходим составной оператор?
- ❷ Напишите программу, которая считывает с клавиатуры  $n$  чисел и затем выводит на экран:
  - a) сумму и среднее арифметическое считанных чисел;
  - б) сумму и среднее арифметическое отрицательных чисел.
- ❸ Напишите программу, которая считывает с клавиатуры  $n$  символов и затем выводит на экран:
  - a) количество считанных десятичных цифр;
  - б) количество нечетных цифр;
  - с) количество считанных букв;
  - д) количество согласных.

Вводимые символы разделяются нажатием клавиши <ENTER>. Предполагается, что будут вводиться десятичные цифры 0, 1, 2, ..., 9 и прописные буквы латинского алфавита A, B, C, ... Z.

## 3.14. Оператор while

Оператор **while** состоит из логического выражения, которое управляет многократным выполнением других операторов. Он имеет следующий вид:

**while** *Логическое выражение* **do** *Оператор*

Примеры:

- 1) **while**  $x > 0$  **do**  $x := x - 1$ ;

- 2) **while**  $x < 3.14$  **do**  
     **begin**  
          $x := x + 0.001$ ;  
         writeln(sin(x));  
     **end**;
- 3) **while** p **do**  
     **begin**  
          $x := x + 0.001$ ;  
          $y := 10 * x$ ;  
          $p := y < 1000$ ;  
     **end**;

Оператор, расположенный после ключевого слова **do**, выполняется многократно до тех пор, пока логическое выражение принимает значение true. Как только логическое значение принимает значение false, оператор, стоящий после ключевого слова **do**, перестает выполняться. Логическое выражение должно иметь наиболее простой вид, так как оно проверяется при каждой итерации.

Обычно оператор **while** используется для организации повторных вычислений в циклах, где управляющей является переменная типа real.

В следующей программе оператор **while** используется для вывода значений функции  $y=2x$ . Аргумент  $x$  принимает значения от  $x_1$  до  $x_2$  с шагом  $\Delta x$ .

```

Program P64;
{ Таблица функции  $y=2*x$  }
var x, y, x1, x2, deltaX : real;
begin
  write('x1='); readln(x1);
  write('x2='); readln(x2);
  write('deltaX='); readln(deltaX);
  writeln('x':10, 'y':20);
  writeln;
  x:=x1;
  while x<=x2 do
    begin
      y:=2*x;
      writeln(x:20, y:20);
      x:=x+deltaX;
    end;
  readln;
end.

```

Оператор **while** особенно полезен в тех случаях, когда трудно определить число повторений некоторой последовательности операторов.

В качестве примера рассмотрим программу P65, которая выводит на экран среднее арифметическое положительных чисел, считываемых с клавиатуры.

```

Program P65;
{ Среднее арифметическое положительных чисел,
  считываемых с клавиатуры }
var x, suma : real;
    n : integer;
begin
  n:=0;
  suma:=0;
  writeln('Введите положительные числа:');
  readln(x);
  while x>0 do
    begin
      n:=n+1;
      suma:=suma+x;
      readln(x);
    end;
  writeln('Было введено ', n, ' положительных чисел. ');
  if n>0 then writeln('среднее=', suma/n)
    else writeln('среднее=*****');
  readln;
end.

```

Отметим, что число повторений составного оператора **begin...end**, входящего в состав оператора **while**, заранее неизвестно. Оператор **while** завершает свое выполнение, когда пользователь вводит число  $x \leq 0$ .

## Вопросы и упражнения

- ❶ Как выполняется оператор **while**?
- ❷ Используя команду **while**, напишите программу, которая выводит на экран значения функции  $y = f(x)$  для аргумента, принимающего значения от  $x_1$  до  $x_2$  с шагом  $\Delta x$ :

a)  $y = \frac{x}{3} + 2;$

b)  $y = \frac{x}{2};$

c)  $y = 3x - 4;$

d)  $y = 4x - 13.$

- ❸ Пользователь вводит с клавиатуры целые положительные числа, разделяемые нажатием клавиши <ENTER>. Признаком конца последовательности является число 0. Напишите программу, которая выводит на экран:
  - a) сумму и среднее арифметическое четных чисел;
  - б) сумму и среднее арифметическое нечетных чисел.
- ❹ Напишите программу, которая выводит на экран значения функции  $y = f(x)$ . Аргумент  $x$  принимает значения от  $x_1$  до  $x_2$  с шагом  $\Delta x$ :

a)  $y = \begin{cases} x, & x > 3; \\ 2x, & x \leq 3; \end{cases}$

b)  $y = \begin{cases} 6x, & x \geq 0; \\ 4x, & x < 0; \end{cases}$

c)  $y = \begin{cases} x + 6, & x > 5; \\ x - 6, & x \leq 5; \end{cases}$

d)  $y = \begin{cases} 3 - x, & x > 4; \\ 3 + x, & x \leq 4. \end{cases}$

Пример:  $y = \begin{cases} x+1, & x > 8; \\ x-2, & x \leq 8. \end{cases}$

```

Program P66;
{ Таблица функции }
var x, y, x1, x2, deltaX : real;
begin
  write('x1='); readln(x1);
  write('x2='); readln(x2);
  write('deltaX='); readln(deltaX);
  writeln('x':10, 'y':20);
  writeln;
  x:=x1;
  while x<=x2 do
    begin
      if x>8 then y:=x+1 else y:=x-2;
      writeln(x:20, y:20);
      x:=x+deltaX;
    end;
  readln;
end.

```

##### 5 Оператор цикла

```
for i:=i1 to i2 do writeln(ord(i))
```

эквивалентен следующей последовательности операторов:

```

i:=i1;
while i<=i2 do
  begin
    writeln(ord(i));
    i:=succ(i);
  end.

```

Напишите эквивалентную последовательность операторов для оператора цикла:

```
for i:=i1 downto i2 do writeln(ord(i))
```

##### 6 Даны описания:

```

var x1, x2, deltaX : real;
    i, n : integer;

```

Какие из следующих последовательностей операторов эквивалентны?

a) 

```

x:=x1;
while x<=x2 do
  begin
    writeln(x);
    x:=x+deltaX;
  end;

```

b) 

```

n:=trunc((x2-x1)/deltaX)+1;
x:=x1;
for i:=1 to n do
  begin
    writeln(x);
    x:=x+deltaX;
  end;

```

```
c) n:=round((x2-x1)/deltaX)+1;
   x:=x1;
   for i:=1 to n do
   begin
     writeln(x);
     x:=x+deltaX;
   end;
```

Аргументируйте свой ответ.

### 3.15. Оператор repeat

Оператор **repeat** указывает на многократное выполнение последовательности операторов в зависимости от значения некоторого логического выражения. Данный оператор имеет вид:

```
repeat
    Оператор_1;
    Оператор_2;
    ...
until <Логическое выражение>
```

Примеры:

```
1) repeat x:=x-1 until x<0;
```

```
2) repeat
    y:=y+delta;
    writeln(y)
until y>20.5;
```

```
3) repeat
    readln(i);
    writeln(odd(i))
until i=0;
```

Операторы, расположенные между ключевыми словами **repeat** и **until**, выполняются многократно до тех пор, пока логическое выражение сохраняет значение **false**. Как только логическое выражение становится истинным, управление переходит к следующему оператору. Очевидно, что операторы, стоящие между ключевыми словами **repeat** и **until**, будут выполнены по крайней мере один раз, так как логическое выражение проверяется лишь после выполнения указанной последовательности операторов.

Как правило, оператор **repeat** используется вместо оператора **while** в тех случаях, когда проверка логического выражения, управляющего повторением, должна производиться после выполнения соответствующей последовательности операторов.

Программа, приведенная ниже, выводит на экран сообщение о четности чисел, считываемых с клавиатуры.

```

Program P67;
{ Четность чисел, считываемых с клавиатуры }
var i : integer;
begin
  writeln('Введите целые числа:');
  repeat
    readln(i);
    if odd(i) then writeln(i:6, ' - нечетное число')
    else writeln(i:6, ' - четное число');
  until i=0;
  readln;
end.

```

Выполнение оператора **repeat** завершается, когда пользователь вводит  $i = 0$ .

Очень часто оператор **repeat** применяется для проверки правильности данных, вводимых с клавиатуры.

Например, предположим, что необходимо написать программу, считывающую с клавиатуры вещественное число  $x$  и выдающую на экран квадратный корень  $y = \sqrt{x}$ . Очевидно, что отрицательные значения переменной  $x$  являются недопустимыми.

```

Program P68;
{ Вычисление квадратного корня }
var x, y : real;
begin
  repeat
    write('Введите неотрицательное число x=');
    readln(x);
  until x>=0;
  y:=sqrt(x);
  writeln('Квадратный корень y=', y);
  readln;
end.

```

При выполнении программы P68 компьютер предлагает пользователю ввести неотрицательное число. В случае, если пользователь по ошибке введет отрицательное число, операторы **write** и **readln** из состава оператора **repeat** будут выполнены заново. Циклический процесс продолжится до тех пор, пока пользователь не введет правильное число.

Из данных примеров видно, что оператор **repeat** используется тогда, когда число повторений некоторой последовательности команд сложно предугадать.

## Вопросы и упражнения

- ❶ Как выполняется оператор **repeat**?
- ❷ Даны следующие операторы:

a) **repeat**  
     <Оператор 1>;  
     <Оператор 2>;  
     ...  
     <Оператор n>;  
**until** p

b) **while not p do**  
**begin**  
     <Оператор 1>;  
     <Оператор 2>;  
     ...  
     <Оператор n>;  
**end.**

Эквивалентны ли эти операторы? Аргументируйте свой ответ.

- ③ Напишите программу, которая считывает с клавиатуры последовательность символов и выводит на экран:

- a) количество считанных десятичных цифр;  
 б) количество четных цифр.

Вводимые символы разделяются с помощью клавиши <ENTER>. Предполагается, что можно вводить десятичные цифры 0, 1, 2, ... 9 и символ \*, который указывает конец последовательности.

- ④ Напишите программу, которая считывает с клавиатуры вещественное число  $x$  и выдает на экран значение выражения  $\frac{1}{x}$ . Очевидно, что значение  $x = 0$  является недопустимым. Для проверки вводимых с клавиатуры данных воспользуйтесь оператором **repeat**.

- ⑤ Эквивалентен ли оператор:

```
for i:=i1 to i2 do writeln(ord(i))
```

последовательности операторов

```
i:=i1;  
repeat  
  writeln(ord(i));  
  i:=succ(i);  
until i>i2;
```

Аргументируйте свой ответ.

- ⑥ Напишите программу, которая выводит на экран значения функции  $y=f(x)$ . Аргумент  $x$  принимает значения от  $x_1$  до  $x_2$  с шагом  $\Delta x$ , а цикл организуется с помощью оператора **repeat**.

a)  $y = 2x;$

c)  $y = x - 4;$

b)  $y = \frac{x}{3} + 9;$

d)  $y = \frac{x}{8} - 6.$

- ⑦ Напишите программу, которая считывает с клавиатуры последовательность символов и выводит на экран:

- a) количество считанных букв;  
 б) количество прописных букв;  
 в) количество строчных букв.

Вводимые символы разделяются с помощью клавиши <ENTER>. Предполагается, что можно вводить строчные и прописные буквы латинского алфавита и символ \*, который указывает конец последовательности.



# РЕДАКТИРОВАНИЕ ИЗОБРАЖЕНИЙ

Изучение содержания этой главы будет основано в большей степени на практических методах обучения. Что это значит? Каждому из вас придется, работая индивидуально или в команде, развивать умение учиться. Это подразумевает обучение с использованием системы поддержки для приложений обработки изображений и различных цифровых учебников, опубликованных в Интернете. Кроме того, практическое обучение предполагает разработку продуктов, содержащих цифровые изображения, и их публикацию в виртуальном пространстве.

## 4.1. Цифровые изображения

Вы уже знаете, что в зависимости от того, как изображения представлены в памяти компьютера, различаем *точечно-ориентированную графику* и *объектно-ориентированную графику*.

### Точечная графика

Напомним, что в точечно-ориентированной графике изображения представлены в памяти компьютера путем разделения на микрозоны, называемые *точками* или *пикселями*. Разложение изображения на пиксели производится с помощью *растра* (от латинского слова *raster*, буквально «грабли»).

В процессе оцифровки изображения пиксели просматриваются в том порядке, в котором читаются: слева направо, сверху вниз. Каждому пикселю соответствует несколько двоичных чисел, которые представляют в форме, понятной компьютеру, информацию о его яркости и цвете. Очевидно, что в точечно-ориентированной графике изображение представлено последовательностью двоичных чисел, и его обработка выполняется посредством операций с этими числами.

Чаще всего растровые изображения используются для обработки визуальной информации, полученной из окружающей реальности. Большая часть оборудования для оцифровки изображений, такого как фотоаппараты, видеокамеры, сканеры и т. д., предоставляет на выходе растровые изображения, в частности файлы в форматах, специально разработанных для представления, хранения и обработки растровых изображений. Растровые изображения также используются для отображения визуальной информации на экранах цифровых устройств и для их печати.

Основное преимущество растровых изображений - их достоверность, то есть аккуратность, точность представления или воспроизведения реальности.

Из недостатков отметим большой объем памяти, необходимый для хранения растровых изображений, и ухудшение качества при увеличении размеров.

### Объектно-ориентированная графика

Чтобы обеспечить лучшее качество цифровых изображений при изменении их размера и уменьшении объема памяти, необходимой для их хранения, в объектно-ориентированной графике изображения состоят из простых графических объектов: линий, квадратов, прямоугольников, кругов, эллипсов и т. д.

В компьютере каждый графический объект в изображении кодируется набором двоичных чисел, обычно называемых *вектором*. Такой набор двоичных чисел содержит всю информацию, необходимую для рисования объекта: координаты центра и радиуса каждого круга, координаты вершин каждого прямоугольника и т. д. Очевидно, что вектор, который характеризует графический объект, также включает информацию о цвете и толщине линий, которые его образуют, цвете заливки, цветовых градиентах и т. д.

Обработка векторных изображений производится путем пересчета координат и размеров каждого графического объекта, содержащегося в изображении, по формулам из аналитической геометрии. Важно знать, что в аналитической геометрии, в отличие от традиционной геометрии, фигуры определяются не изображениями, а с помощью формул, и их преобразование чисто алгебраическое. Для этого плоскость, в случае двумерных изображений, и пространство, в случае трехмерных изображений, снабжены системами координат, обычно декартовыми.

Векторные изображения можно увеличивать и уменьшать без ухудшения их качества, пересчитывая их размеры в соответствии с математическими формулами, связанными с каждым графическим объектом. Более того, с помощью расчетов графические объекты можно анимировать, перемещать в пространстве, перекрашивать, трансформировать и т.д., что особенно важно при создании цифровых симуляторов и компьютерных игр. Очевидно, что в случае приложений цифровой графики компоненты аналитической геометрии, на которых основана обработка векторных изображений, то есть формулы и вычислительные алгоритмы, «невидимы» для пользователя, он оперирует только терминами, характерными для графического дизайна.

Недостатком векторных изображений является то, что кодирование сложной визуальной информации простыми графическими объектами, описываемыми с помощью математических формул, приводит к снижению достоверности представления или воспроизведения реального мира. Очевидно, что снижение точности вызвано не самими математическими формулами, а сложностью и большим количеством объектов, которые могут потребоваться для обеспечения желаемого уровня точности. Однако с увеличением производительности современных компьютеров сложность и количество графических объектов перестают быть непреодолимым препятствием, что можно легко наблюдать в случае компьютерных игр, изображения в которых становятся более сложными и близкими к таким, какими мы видим их в реальном мире.

Современные приложения для обработки изображений позволяют группировать объекты, тем самым давая пользователю возможность создавать сложные объекты из простых графических объектов.

## Преобразование изображений

Растровые изображения можно преобразовывать в векторные изображения и наоборот, векторные изображения - в растровые. Более того, современные приложения для обработки изображений позволяют создавать и обрабатывать смешанные цифровые изображения, т. е. изображения, содержащие как растровые, так и векторные компоненты.

Обычно преобразование растровых изображений в векторные производится для того, чтобы уменьшить занимаемый ими объем, объединить реальный мир с элементами виртуального мира (дополненная реальность), распознавать формы. Преобразование векторных изображений в растровые производится с целью их отображения на экранах цифровой техники и для печати. Для таких преобразований используются следующие термины: *растеризация* (от слова «*raster*») и *рендеринг* (от английского слова «*rendering*» - визуализация, выражение в визуальной форме).

## 4.2. Цветовые модели

Для захвата и воспроизведения изображений с помощью цифрового оборудования используется несколько цветовых моделей:

**Монохромная модель**, обычно черно-белая, в которой цифровое изображение содержит в каждой из микрозон только оттенки серого. В большинстве случаев эти оттенки кодируются двоичными числами, состоящими из восьми бит.

Количество информации в таком изображении определяется по формуле:

$$I = X Y \text{ (бит)},$$

где  $X$  и  $Y$  – размеры изображения в пикселях.

**Модель RGB.** Эта модель используется для визуализации изображений с помощью светового излучения. Человеческий глаз, улавливая свет, излучаемый источником изображения, воспринимает цвет, «собирая» три составляющих цвета: красный (*Red*), зеленый (*Green*) и синий (*Blue*).

Яркость и цвет каждой из микрозон кодируются пропорциями каждого из трех основных цветов. Таким образом, в цифровом изображении каждой микрозоне соответствуют три байта, первый из которых представляет оттенки красного, второй - оттенки зеленого, а третий - оттенки синего.

Количество информации в цветном изображении определяется по формуле:

$$I = 3 X Y \text{ (бит)}.$$

Модель *RGB* используется для изображений, которые выводятся на экраны дисплеев и мультимедийных проекторов. Напоминаем, что в этих устройствах каждый пиксель матрицы, формирующей изображение, состоит из трех люминесцентных ячеек, одна из которых излучает красный цвет, другая – зеленый цвет, а третья – желтый цвет.

Также модель *RGB* используется в фотоаппаратах, видеокамерах, сканерах и т. д. В этих устройствах каждый пиксель матрицы захвата изображения состоит

из трех светочувствительных ячеек, одна из которых чувствительна только к красному свету, другая – только к зеленому свету, а третья – только к желтому свету.

Например, на *рисунке 4.1* показано диалоговое окно **Edit color** (Редактировать цвет) приложения **Paint 3D**, предназначенное для выбора желаемого цвета на основе модели *RGB*.

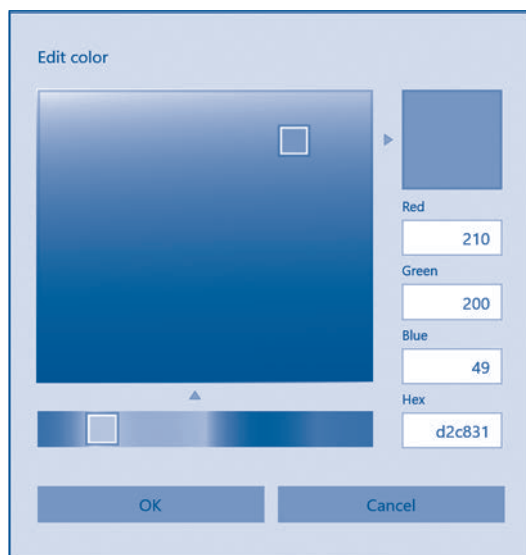


Рис. 4.1. Выбор желаемого цвета на основе модели *RGB*

**Модуль СМУ.** Эта модель используется для изображений, выводимых на печать. Напомним, напечатанные изображения не излучают, а только отражают падающий на них свет. Как следствие, весь цветовой спектр получается путем смешивания красок трех базовых цветов: голубой (*Cyan*), пурпурный (*Magenta*) и желтый (*Yellow*).

Так как для получения черного цвета необходима комбинация всех трех цветов, а они дорогие, в печатающих устройствах используется и черный цвет. Как следствие, графические редакторы предоставляют пользователям и цветовую модель *СМУК*, в которой дополнительно к цветам модели *СМУ* используют четвертый цвет – черный (*black*).

Например, в приложении **Gimp** пользователь может выбрать нужный цвет с помощью курсоров, которые фиксируют пропорцию каждого из цветов *СМУК* (рис. 4.2). Отметим, что соответствующие пропорции указываются в процентах.

**Модели основанные на палитрах цветов.** Визуально палитра цветов представляет собой последовательность цветных прямоугольников, пронумерованных, например, от 0 до 255. Подобные палитры часто встречаются в индустрии красоты, где цвет краски для волос или губной помады указывается не только с помощью рисунка, но и числовым значением. Цветовые палитры широко используются при ремонте кузовов автомобилей, при этом каждый производитель автомобилей предлагает ремонтным мастерским свои палитры. Также цветовые палитры широко используются в полиграфии, в текстильной промышленности, в лакокрасочной промышленности, в техническом дизайне и т. д.



Рис. 4.2. Выбор желаемого цвета на основе CMYK

В памяти компьютера цветовые палитры представлены кодами цветов соответствующих прямоугольников в соответствии с моделью RGB или CMY и присвоенными им номерами. Очевидно, что соответствующие коды и цифры «невидимы» для пользователя, он работает на экране только с актуальными цветами (рис. 4.3).

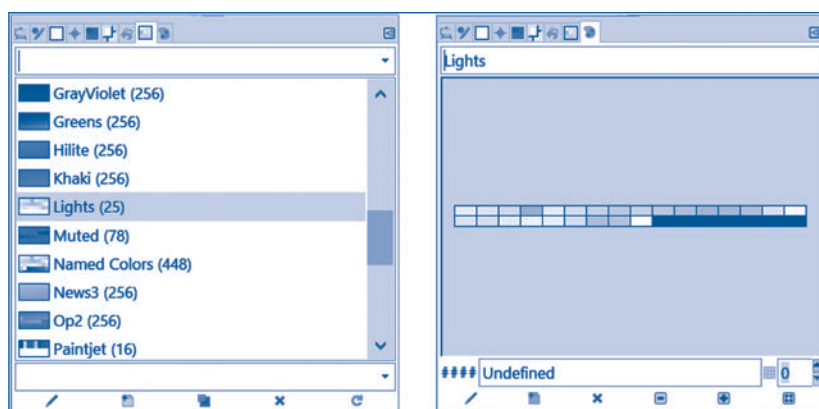


Рис. 4.3. Палитра цветов **Lights** (Раскраска) в графическом редакторе **Gimp**

Когда изображения, подлежащие обработке, содержат относительно небольшое количество цветов, использование палитр уменьшает объем памяти, необходимый для хранения этих изображений. В качестве примеров приведем дорожные знаки и разметку, рекламные щиты, афиши, плакаты, буклеты, схемы различных технических устройств и т. д.

### 4.3. Форматы графических файлов

В ходе развития информационных и коммуникационных технологий были разработаны разные форматы графических файлов. Разработка новых графических форматов направлена на уменьшение размера файлов, обеспече-

ние качества изображений и возможности воспроизведения их на различных устройствах.

### Сжатие цифровых изображений

Количество информации в изображениях намного больше, чем в тексте. Например, изображения, отображаемые на экранах мониторов последнего поколения, имеют размеры 8192×4320 пикселей. Количество информации в таком изображении составляет:

$$I = 3 \times 8192 \times 4320 = 106168320 \text{ байт} \approx 101 \text{ Мегабайт}.$$

Светочувствительные матрицы профессиональных цифровых фотоаппаратов имеют размеры 38000×38000 пикселей. Количество информации в изображении, захваченном такой матрицей, составляет:

$$I = 3 \times 38000 \times 38000 \approx 4131 \text{ Мегабайтов} \approx 4,0 \text{ Гигабайта}.$$

Чтобы уменьшить объем памяти, необходимый для хранения цифровых изображений, их сжимают. Для этого разработаны специальные алгоритмы, которые изучаются в расширенных курсах по информатике. Эти алгоритмы могут обеспечить сжатие цифровых изображений в десятки и даже сотни раз с потерей графической информации или без потери.

Алгоритмы сжатия цифровых изображений *без потери информации* основаны на том, что на многих изображениях присутствуют идентичные соседние микрозоны. Информация о каждой из этих микрозон может быть заменена информацией только об одной из них.

Алгоритмы сжатия цифровых изображений *с потерей информации* уменьшают количество цветов в сжатых изображениях и удаляют из них очень мелкие детали. В некоторых случаях эти преобразования не воспринимаются человеческим глазом.

### Форматы растровых изображений

Основными форматами графических файлов для растровых изображений являются:

**ВМР** – битовая карта (англ. *bitmap*). Этот формат используется для хранения значений элементов (микрозон) изображений, каждой микрозоне соответствует от одного (черно-белое изображение) до 64 бит (изображения с очень широким диапазоном цветов). Формат обеспечивает очень хорошее качество изображения, однако размеры соответствующих файлов очень большие. Файлы в этом формате не рекомендуется отправлять через Интернет.

Расширения файлов для этого формата: **.bmp, .dib, .rle**.

**JPEG** – аббревиатура от *Joint Photographic Experts Group* (Объединенная группа экспертов по фотографии). Этот формат позволяет сжимать растровые изображения с потерями или без них. В случае сжатия без потерь размер файла уменьшается почти вдвое, а в случае сжатия с потерями – до 80 раз. Рекомендуется для хранения фотографий и меньше для технических чертежей. Он широко используется в Интернете.

Расширения файлов для этого формата: **.jpg, .jfif, .jpe** или **.jpeg**, наиболее популярным из которых является **.jpg**.



**TIFF** – это аббревиатура от *Tagged Image File Format* (Теговый формат файла изображения). Формат используется для хранения растровых изображений с очень широкой цветовой гаммой, особенно в полиграфии. Обеспечивает сжатие без потерь или с потерями.

Расширения файлов для этого формата: **.tiff** или **.tif**.

**GIF** – аббревиатура от *Graphics Interchange Format* (Формат обмена графикой). Он использует 256 цветов и обеспечивает сжатие без потерь. Очень долгое время это был самый распространенный растровый формат в Интернете. Он очень хорош для представления схем, логотипов, печатей, текстов на рисунках и реже для фотографий.

Файлы, хранящиеся в этом формате, имеют расширение **.gif**.

**PNG** – аббревиатура от *Portable Network Graphics* (Переносимая сетевая графика). Формат предназначен для хранения растровых изображений, сжатых без потерь. Он был разработан, чтобы заменить формат **GIF** и частично формат **TIFF**. Формат широко используется в Интернете, соответствующие файлы имеют расширение **.png**.

### **Форматы векторных изображений**

В случае векторных изображений основными форматами в настоящее время являются:

**WMF** – аббревиатура от *Windows Metafile* (Метафайл Windows).

**SVG** – аббревиатура от (*Scalable Vector Graphics*). (Масштабируемая векторная графика).

**EPS** – аббревиатура от *Encapsulated PostScript* (Инкапсулированный PostScript). Язык PostScript широко используется в редакциях.

Иногда форматы векторных изображений содержат префикс **2D** или **3D**, указывающий на то, что изображение двухмерное или трехмерное.

### **Форматы смешанных документов**

**PDF** – аббревиатура от *Portable Document Format* (Формат переносимого документа). Первоначально формат был разработан для распространения документов в электронном виде, особенно тех, которые предназначены для типографий. С тех пор он получил широкое распространение в Интернете, поскольку не зависит от специфики операционных систем и периферийного оборудования, используемого для печати или просмотра этих документов. Позволяет вставлять тексты, растровые изображения и векторные изображения в один документ.

**CGM** – аббревиатура от *Computer Graphics Metafile* (Метафайл компьютерной графики). Формат в основном используется в области автоматизированного проектирования различных технических систем и в меньшей степени в цифровом искусстве или в Интернете.

Обычно в приложениях для обработки изображений пользователь выбирает формат графического файла с помощью специальных диалоговых окон (рис. 4.4).

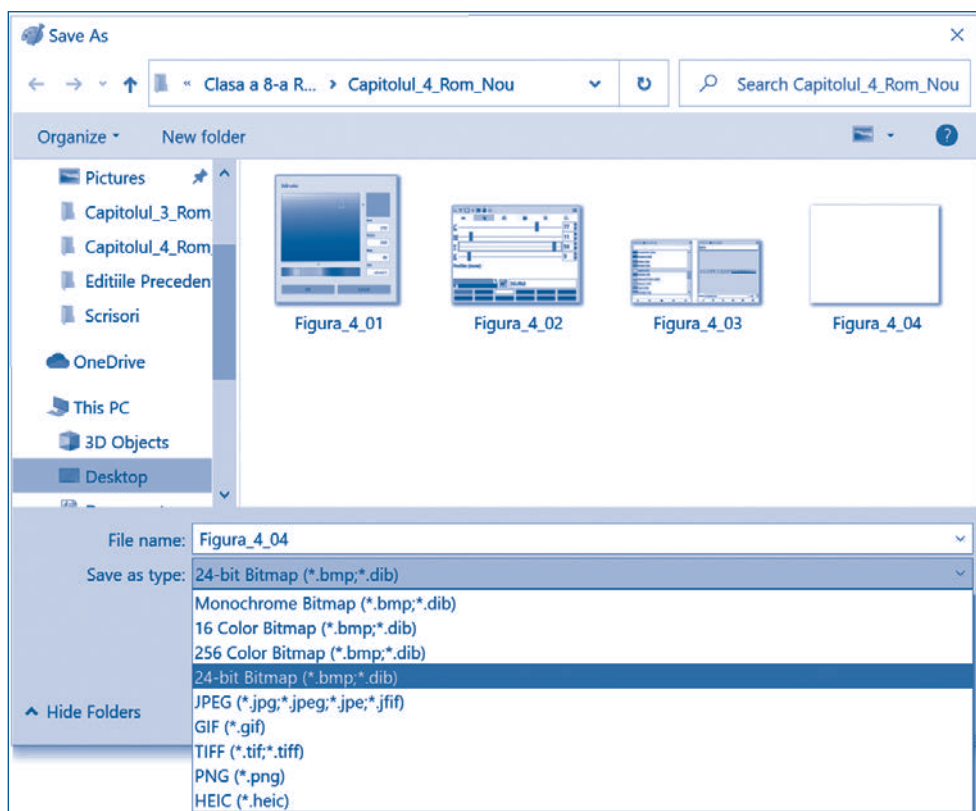


Рис. 4.4. Выбор формата графического файла

Отметим, что многие форматы графических файлов, обычно используемые в компьютерной графике, принадлежат крупным компаниям, специализирующимся на производстве оборудования и приложений для цифровой обработки изображений.

## 4.4. Практические занятия: оборудование и форматы графических файлов

Для закрепления теоретических знаний, полученных в процессе изучения предыдущих параграфов, рекомендуем следующие практические занятия.

1. Повторите темы, связанные с квантованием изображения: микрозона, точка, пиксель, растр, разрешение, цифровые изображения.

2. **ЭКСПЕРИМЕНТИРУЙТЕ!** Узнайте технические параметры имеющихся у вас фотоаппаратов: цифровых фотоаппаратов, встроенных в мобильные телефоны и планшеты фотокамер. Узнайте об используемых в них форматах графических файлов.



3. **ИССЛЕДУЙТЕ!** Вычислите приблизительно размер и количество информации в изображениях, снятых вашими фотокамерами.

4. **ЭКСПЕРИМЕНТИРУЙТЕ!** Узнайте технические параметры ваших сканеров, плоттеров и цветных принтеров. Узнайте об используемых в них форматах графических файлов.

5. **ИССЛЕДУЙТЕ!** Выберите из имеющейся у вас типографской продукции (руководства, словари с картинками, фотоальбомы, буклеты, флаеры и т. д.) несколько изображений. Оцените количество информации в каждом из этих изображений.

6. **СОЗДАЙТЕ!** Напишите небольшое эссе об эволюции форматов графических файлов.

7. **ПРАКТИЧЕСКОЕ ИССЛЕДОВАНИЕ.** Проведите сравнительный анализ форматов графических файлов. С помощью графического редактора сохраните одно и то же изображение в разных форматах. Определите связь между качеством цифровых изображений, хранящихся в различных графических форматах, и объемом памяти, которую они занимают.

## 4.5. Приложение *Paint 3D*

В предыдущих классах вы изучали, а потом часто использовали приложение **Paint**. Это простое приложение, разработанное с появлением первых операционных систем семейства **Windows**, оказалось очень полезным для базовой обработки двумерных изображений: вставки и вырезания фрагментов, поворота и изменения размера, заливки цветом, вставки коротких текстов и т. д.

С развитием компьютерной графики для начинающих пользователей был разработан новый графический редактор **Paint 3D**. Суффикс **3D** в названии этого приложения указывает на то, что с его помощью можно создавать и обрабатывать не только двумерные изображения, но и трехмерные. Окно приложения **Paint 3D** показано на *рис. 4.5*.

Взаимодействие пользователя с приложением **Paint 3D** осуществляется с помощью элементов управления, характерных для современных графических интерфейсов: пиктограмм, кнопок, диалоговых окон, панелей, контекстных меню и т. д. Таким образом, пиктограммы в верхней части окна имеют следующее значение:

**Menu** (Меню) – содержит команды для открытия и сохранения графических файлов. В случае открытия двумерных изображений (форматы BMP, JPG, PNG и т.д.) растровое изображение будет служить фоном. Любое выделение на этом изображении можно преобразовать в трехмерный графический объект с помощью команды **Make 3D** (Сделать 3D) в контекстном меню. Очевидно, что с растровыми изображениями можно выполнять все традиционные операции: вставку, обрезку, поворот, изменение размера, заливку цветом, вставку текста и т.д.

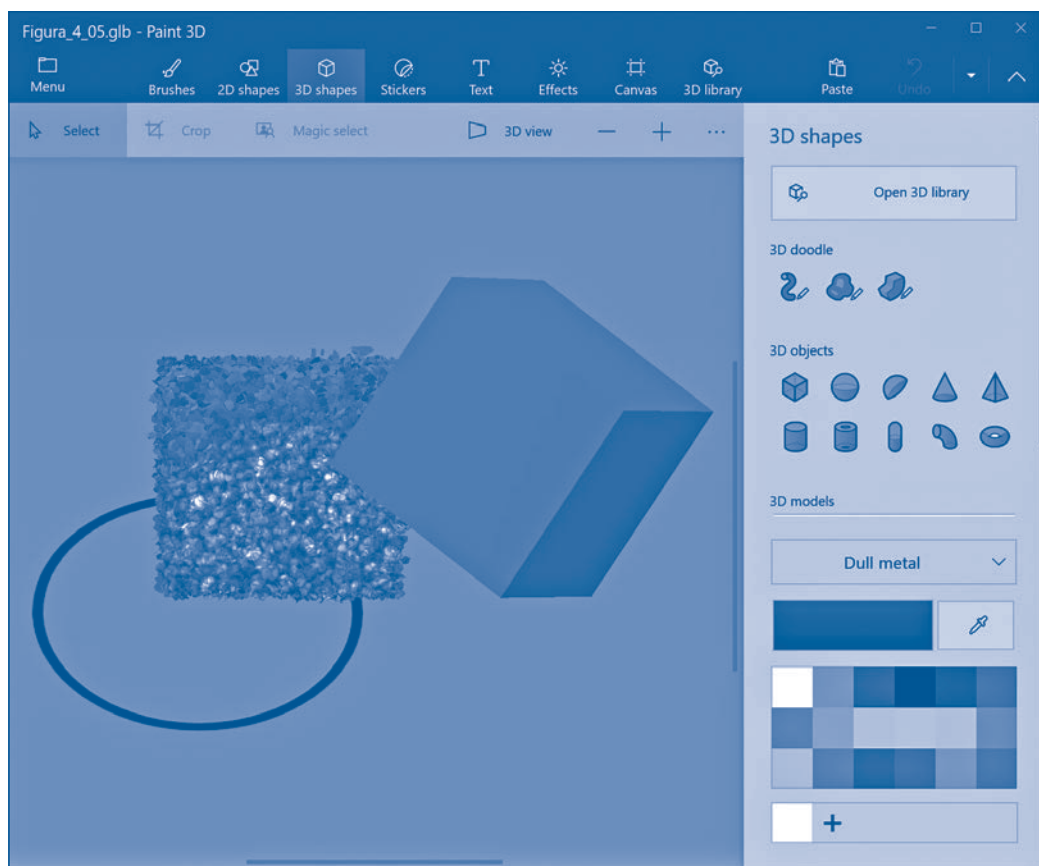


Рис. 4.5. Окно приложения **Paint 3D**

Изображения, созданные с помощью **Paint 3D**, можно сохранять в форматах двухмерных изображений (параметр **Image** (Изображение) в меню **Save As** (Сохранить как), в форматах 3D-моделей (параметр **3D model**) или как проект (параметр **Paint 3D project**). Отметим, что для хранения трехмерных моделей разработаны специальные графические форматы, основными из которых являются **GLB**, **FBX**, **3MF**).

Чтобы индивидуально или в команде изучить, как использовать приложение **Paint 3D**, в меню есть специальная команда **Learn and Feedback** (Обучение и обратная связь), которая обеспечивает доступ к набору онлайн-руководств.

**Brushes** (Кисти) - панель, отображаемая сразу после запуска этой команды, содержит богатый набор кистей и цветовую палитру. Пользователь также может выбрать внешний вид, который будет присвоен изображаемому объекту, например матовый, глянцевый, матовый металл, полированный металл.

**2D shapes** (двухмерные формы) - панель содержит заранее нарисованные фигуры, которые можно вставить в изображение как графические объекты. Как и в случае любого другого графического объекта, у пользователя есть возможность задать их свойства (толщину и цвет линии, цвет заливки), изменить их размер и повернуть.

**3D shapes** (Трехмерные формы) - содержит трехмерные фигуры. В дополнение к уже нарисованным фигурам на панели также есть инструменты для рисования фигур с острыми и закругленными краями. Эти инструменты сгруппированы под названием **Doodle**. Первоначальное значение этого английского слова - «каракули», но в случае графических редакторов его следует переводить как «рисование от руки».

Команда **Open 3D library** (Открыть библиотеку 3D) из этого меню предоставляет доступ к богатой коллекции моделей, хранящихся на специальном сервере Корпорации Microsoft (рис 4.6).

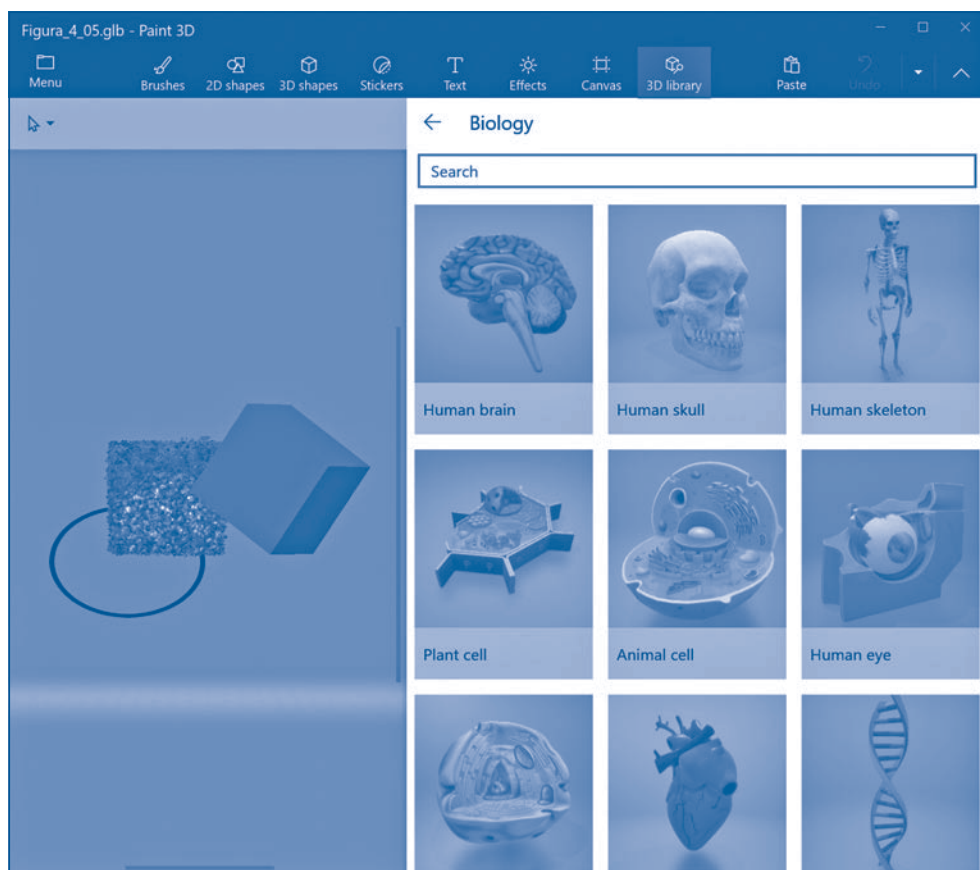


Рис. 4.6. Библиотека 3D, категория Биология

Модели в этой библиотеке сгруппированы по категориям, основными из которых являются *Животные*, *Космическое пространство*, *Динозавры*, *Цветы и растения*, *Биология* и другие. Очевидно, что пользователь может вставлять эти объекты в свои собственные графические работы, изменяя их внешний вид в соответствии с их информационным смыслом.

**Stickers** (Наклейки, этикетки) - команда открывает доступ к панели, держащей богатую коллекцию небольших этикеток, которые, прикрепляясь к графическому объекту, становятся частью его поверхности (рис. 4.7).

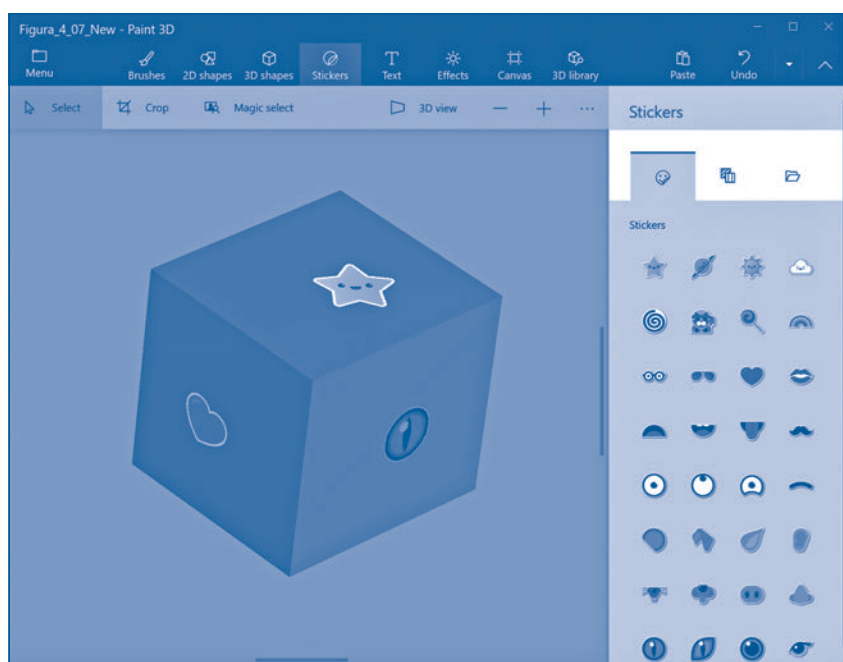


Рис. 4.7. Вставка наклеек

В дополнение к заранее нарисованным этикеткам панель **Stickers** содержит страницу **Textures** (Текстуры), которая придает выбранной поверхности определенный вид, например дерево, живая изгородь, мех, галька, бетон, песок и т. д. (рис. 4.8).

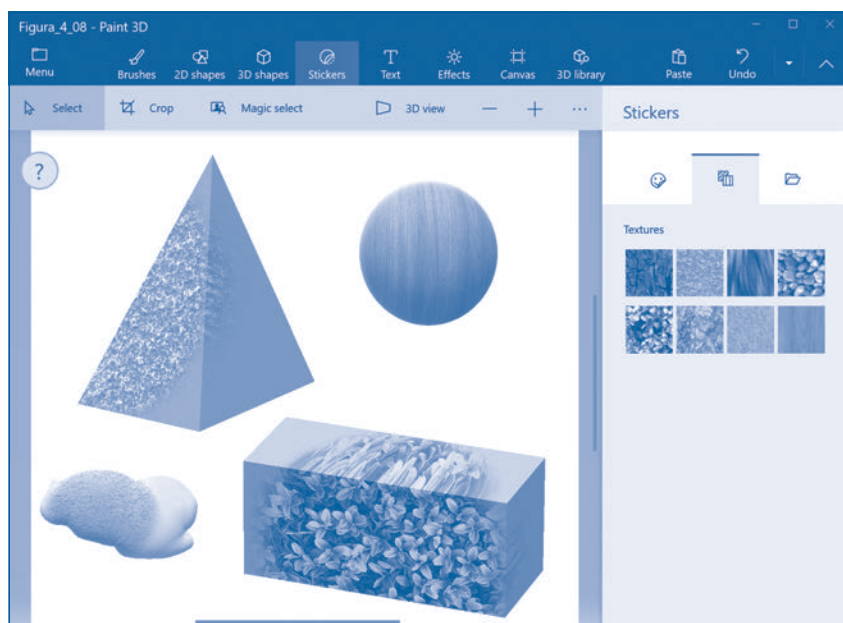


Рис. 4.8. Применение текстур

**Effects** (Эффекты) - команда отображает на экране панель, в которой пользователь может выбирать фильтры и положение источника света.

На панели инструментов отметим кнопку **Magic Select**, позволяющую выделить из фонового изображения ту часть, которую желательно удалить. Удаляемый фрагмент преобразуется в графический объект, а свободное пространство автоматически заполняется изображением, созданным искусственным интеллектом приложения **Paint 3D**.

Кнопка 3D просмотра (**3D view**) на панели инструментов дает пользователю возможность просматривать трехмерные объекты под любым углом, вращать их, изменять их размеры и положение.

Обратите внимание, что некоторые команды и меню в приложении **Paint 3D**, например меню **Text** (Текст), аналогичны командам и меню в приложении **Paint** или объектно-ориентированной графике офисных приложений **Word** и **PowerPoint**.

## 4.6. Практические занятия: Обработка трехмерных изображений

1. Создайте в области рисования изображения, показанные на *рисунках 4.5, 4.6 и 4.7*.

2. Загрузите на свой компьютер изображения, снятые с помощью цифровых фотокамер. Вставьте в эти изображения короткие тексты, описывающие их тему, место, где они были сделаны, дату и время. Придайте этим текстам художественный вид.

3. Вставьте на изображения, снятые цифровыми камерами, наклейки, отражающие смысл изображений.

4. **РАЗРАБОТАЙТЕ!** Используя библиотеку трехмерных объектов, создайте небольшие карточки для реальных дисциплин, которые вы изучаете: математики, физики, биологии, химии, информатики.

5. **СОЗДАЙТЕ!** На основе библиотек объектов сделайте несколько иллюстраций самых увлекательных художественных произведений из области фантастики, которые вы читали.

6. **ИНДИВИДУАЛЬНОЕ ИЛИ КОМАНДНОЕ ОБУЧЕНИЕ.** Просмотрите видеоруководства, к которым можно получить доступ с помощью команды **Learn and Feedback** в меню приложения **Paint 3D**. Следите за публикациями **Paint 3D** в блоге **Windows**.

7. **УПРАЖНЯЙТЕСЬ!** Используя интерактивные руководства в Интернете, например, предоставляемые службой хранения аудио и видео **YouTube**, узнайте назначение и способы использования каждой из команд в **Paint 3D**. Создайте изображения, аналогичные тем, которые показаны в соответствующих руководствах.

8. **ИССЛЕДУЙТЕ!** Найдите в Интернете трехмерные модели объектов. Оцените возможности их использования в изображениях, которые вы собираетесь создать. Оцените их художественную ценность и соответствие информационному смыслу, который вы собираетесь передать.

**9. УЧИТЕСЬ УЧИТЬСЯ! 3D Viewer** - это приложение, которое используется для просмотра и редактирования трехмерных изображений в операционной системе **Windows**. Помимо собственно просмотра, приложение позволяет пользователю управлять источниками освещения и изменять тени трехмерных изображений, применять различные текстуры. Более того, оно позволяет создавать изображения из категории дополненной реальности.

Используя интерактивную справочную систему и руководства, подробно изучите меню и команды в этом приложении. Ознакомьтесь с богатой библиотекой 3D-моделей.

**10. ЭКСПЕРИМЕНТИРУЙТЕ!** Трехмерные объекты можно материализовать (распечатать) с помощью **3D-принтеров**. Если в вашей школе есть такой принтер, распечатайте некоторые трехмерные объекты, созданные с помощью приложения **Paint 3D**.

**11. ИССЛЕДУЙТЕ!** Разработайте небольшое исследование услуг 3D-печати, доступных в нашей стране. Узнайте, как стоимость этих услуг зависит от технологии 3D-печати и сложности печатаемых объектов.

**12. ИНТЕГРИРУЙТЕ!** Приложение **Paint 3D** очень полезно для разработки проектов **STEAM** (*Science, Technology, Engineering, Arts and Mathematics* - Наука, технология, инженерия, искусство и математика). Эти проекты основаны на интеграции знаний из соответствующих областей с помощью ИТ-инструментов. Сервис хранения аудио и видео **YouTube** содержит некоторое количество руководств по созданию 3D-проектов, посвященных интеграции знаний. Изучите эти руководства и разработайте свой проект **STEAM**.

## 4.7. Слои и каналы цветов

**Слои.** Расширенная обработка цифровых изображений предполагает использование слоев. Каждый слой содержит свое изображение, а результирующее изображение, которое видит пользователь, формируется путем их наложения (рис. 4.9).

Очевидно, что изображение на слое, расположенном ниже, будет видно, только если изображения на слоях выше будут частично или полностью прозрачными. Другими словами, слой можно интерпретировать как стеклянную пластину с определенными изображениями на ней.

Когда некоторые фрагменты изображений на одном слое непрозрачны, они будут маскировать соответствующие части изображений, поступающих из нижележащих слоев. Поэтому при использовании слоев к описанию каждого пикселя, помимо байтов, указывающих пропорции базовых цветов, добавляется еще один байт. Этот байт указывает степень непрозрачности, обычно выраженную в процентах.

Таким образом, степень непрозрачности 0% соответствует полной прозрачности, а изображение на соответствующем слое абсолютно невидимо. В этом случае изображение из нижележащего слоя будет полностью видимым.



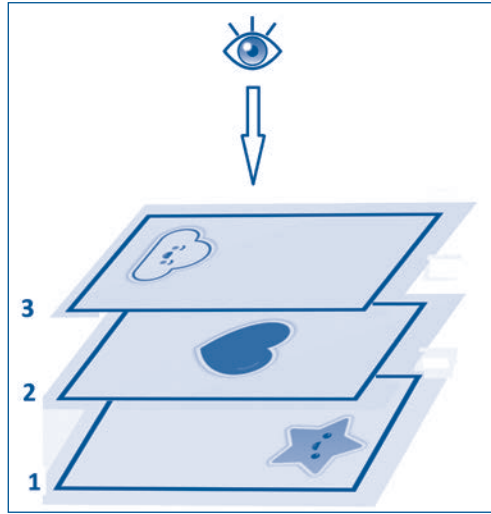


Рис. 4.9. Формирование изображения путем наложения слоев

100% непрозрачности соответствует полной непрозрачности: изображение, находящееся на слое ниже, полностью заслоняется (закрывается) изображением на текущем слое.

В случае степени непрозрачности с промежуточными значениями от 0% до 100% изображение на текущем слое будет перекрывать изображение, исходящее из слоя ниже, и по мере увеличения степени непрозрачности изображение в текущем слое будет закрывать все больше и больше изображение, которое приходит из нижележащего слоя.

Например, на *рисунке 4.10* показаны изображения, которые видит пользователь в зависимости от степени непрозрачности слоев, из которых он сформирован.

Уровень непрозрачности (Ораque) слоев			Результирующее изображение	
1	2	3		
★ 100%	+ ♥ 100%	+ ☁ 100%	=	☁
★ 100%	+ ♥ 100%	+ ☁ 0%	=	♥
★ 100%	+ ♥ 0%	+ ☁ 0%	=	★
★ 100%	+ ♥ 90%	+ ☁ 90%	=	★ ♥ ☁

Рис. 4.10. Влияние непрозрачности на результирующее изображение

Если верхний слой 3 полностью непрозрачен, пользователь видит только изображение на нем, а изображения на других слоях затмеваются. Когда слой 3 полностью прозрачен, пользователь видит изображение, поступающее со слоя 2. Поскольку этот слой полностью непрозрачен, изображение, поступающее со слоя 1, затмевается.

Если слои 2 и 3 полностью прозрачны, пользователь видит изображение только на слое 1.

Если слои 2 и 3 частично прозрачны, пользователь видит изображение, полученное путем наложения изображений на всех трех слоях. В итоговом изображении наиболее ярко выражено изображение на уровне 3, за которым следует изображение на уровне 2, а самым слабым будет изображение на уровне 1.

В целом использование слоев позволяет не только простое наложение нескольких изображений, но и выборочное применение эффектов, маскирование определенных фрагментов изображений, исправление любых дефектов, например «красных глаз» на фотографиях, сделанных со вспышкой.

**Цветовые каналы.** Концептуально цветовой канал представляет собой серию чисел: по одному для каждого пикселя, составляющего изображение. Число, прикрепленное к каждому из пикселей, указывает долю цвета, связанную с каналом, при формировании внешнего вида пикселя.

При загрузке изображения графические редакторы автоматически создают три канала со смысловыми названиями *Red* (Красный), *Green* (Зеленый) и *Blue* (Синий), которые соответствуют основным цветам, соответственно красному, зеленому и синему. Если изображение состоит из слоев, графические редакторы автоматически создают еще один канал, называемый *Alpha*. Этот канал содержит информацию о прозрачности / непрозрачности каждого пикселя обрабатываемого изображения.

В дополнение к красному, зеленому, синему и альфа-каналам, автоматически создаваемым графическими редакторами, дизайнер также может создавать свои собственные каналы, ассоциируя их с желаемыми цветами.

Например, на *рисунке 4.11* показаны диалоговые окна графического редактора *GIMP*, предназначенные для работы с цветовыми каналами.

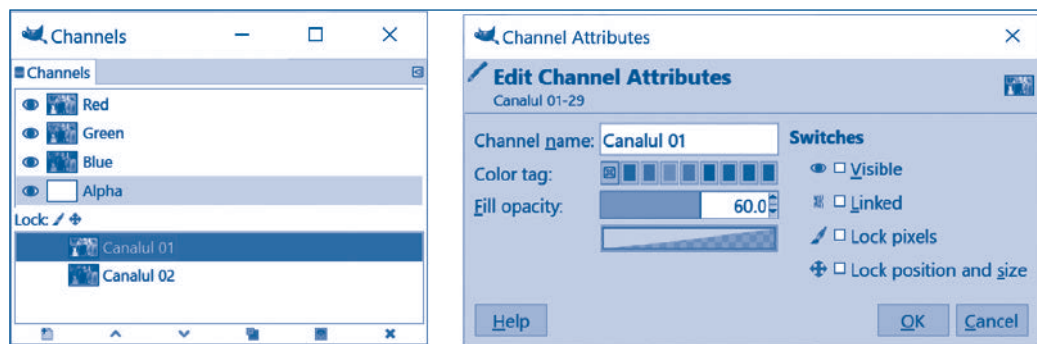


Рис. 4.11. Диалоговые окна для работы с цветовыми каналами

Окно **Channels** (Каналы) содержит автоматически создаваемые каналы **Red**, **Green**, **Blue** и **Alpha**, а также каналы, созданные пользователем **Canalul 01**



и **Canalul 02**. Окно **Channel Attributes** (Атрибуты канала) содержит элементы управления, которые позволяют устанавливать и изменять свойства первого из созданных пользователем каналов.

Включение или отключение канала приводит к включению или исключению этого цвета из пиксельного отображения полученных изображений. Активируя или деактивируя определенные каналы и изменяя их свойства, дизайнер может корректировать цвета обрабатываемых изображений, может создавать маски для включения, исключения, акцентирования или размытия определенных фрагментов этих изображений.

## 4.8. Приложение GIMP

Графический редактор **GIMP** (*General Image Manipulation Program* – Общая программа обработки изображений) – это бесплатное приложение, разработанное группой добровольцев.

Приложение **GIMP** может работать в одном или нескольких окнах. Обычно при работе в нескольких окнах обрабатываемое изображение отображается в главном окне, самом большом, в то время как другие окна, называемые *диалоговыми* окнами, содержат элементы управления, необходимые для выполнения различных задач. Диалоговые окна можно перетаскивать на экране монитора, что позволяет дизайнеру максимально эффективно использовать рабочий стол операционной системы. В случае компьютерной графики это очень важно, поскольку соответствующие редакторы содержат сотни таких окон, а выполнение даже одной задачи требует одновременного отображения на экране 3-4 и более диалоговых окон.

Главное окно приложения **GIMP** (рис. 4.12) содержит, сверху вниз, строку с именем обрабатываемого файла, строку меню, область обработки изображений и строку состояния. Пользователь настраивает свое рабочее пространство и обрабатывает изображения, используя команды из меню в строке меню и команды из контекстных меню.

Меню в верхней части окна приложения **GIMP** имеют следующий смысл:

**File** – содержит команды открытия, сохранения, закрытия графических файлов. Графический файл может быть открыт как изображение или как слой. В последнем случае изображение из файла будет включено как отдельный слой обрабатываемого изображения.

**Edit** – содержит несколько команд для редактирования изображений, таких как обрезка, копирование, вставка и т. д. Также в этом меню есть команда **Preferences** (Настройки), которая позволяет настроить рабочее пространство.

**Select** – содержит различные инструменты для выбора, а также для сохранения выбранного содержимого.

**View** – команды в этом меню позволяют выбрать как режимы отображения изображений, подлежащих обработке, так и отображение / скрытие диалоговых окон.

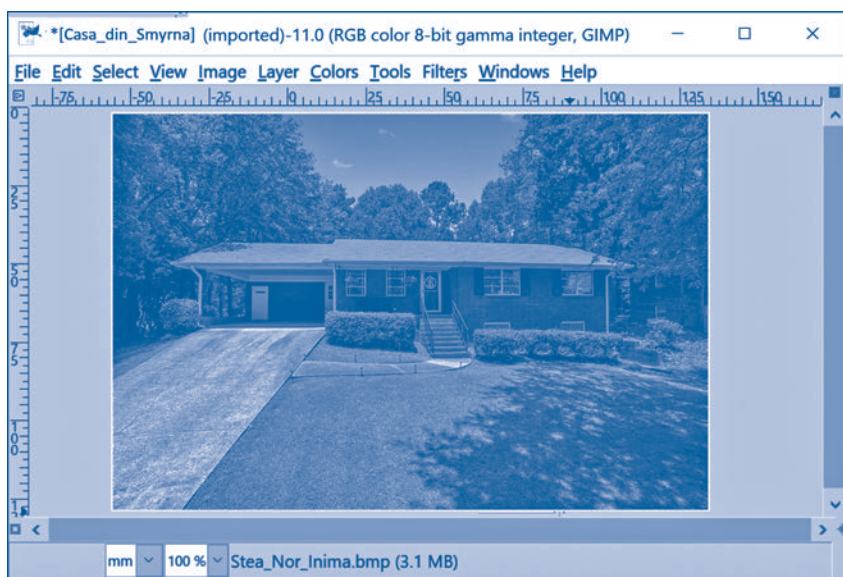


Рис. 4.12. Окно приложения **GIMP**

**Image** – включает список команд для установки общих свойств изображений и выполнения операций, часто используемых в процессе обработки изображений: канва, выравнивание сетки, поворот, зеркальное отображение и т. д.

**Layer** – в этом меню сгруппированы основные команды работы со слоями, составляющими обрабатываемое изображение: создание, удаление, дублирование слоев, изменение порядка слоев, установка непрозрачности / прозрачности, создание масок, объединение слоев и т. д.

**Colors** – включает команды обработки цвета. Среди них упомянем команды, связанные с регулировкой яркости и контрастности, уменьшением или увеличением количества цветов, приближением цветов изображения к наиболее близким в определенной цветовой палитре.

**Tools** – содержит список команд для обработки изображений, сгруппированных по категориям: выделение, рисование, преобразование изображения, вставка текста и т. д. Пользователь может настроить рабочую среду, создав так называемые панели инструментов, в которых он может группировать пиктограммы часто используемых инструментов. Эти панели могут постоянно отображаться на экране либо в одном окне вместе с обрабатываемым изображением, либо в отдельных окнах.

**Filters** – содержит несколько фильтров, которые, будучи применимы к обрабатываемому изображению, изменяют его внешний вид: исправление «красных глаз», размытие, искажение, повышение четкости, применение света и тени, включая перспективу и многое другое.

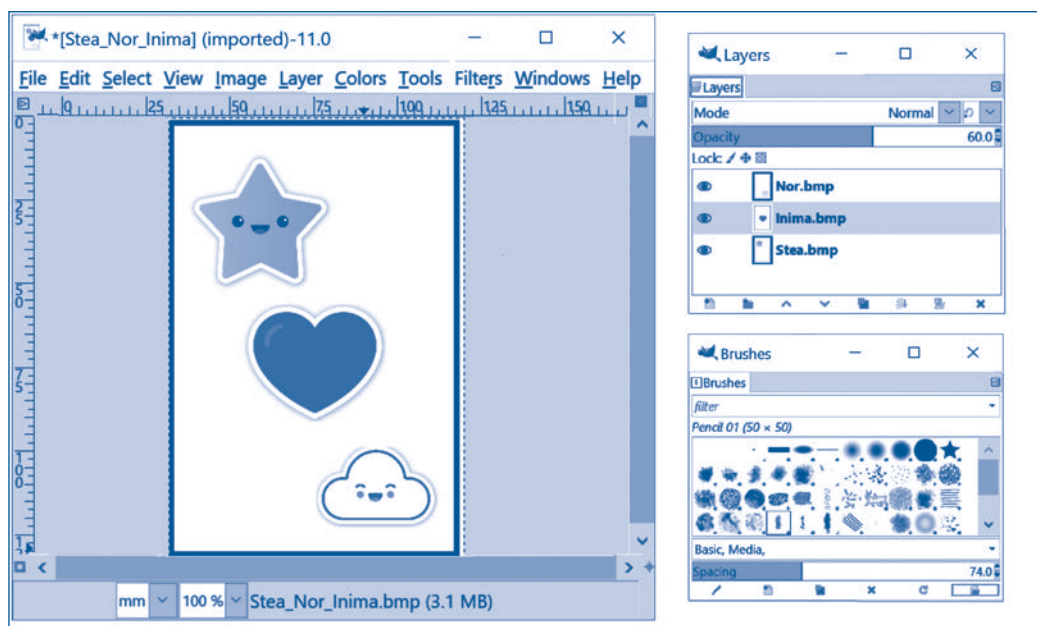
**Windows** – включает команды для управления окнами. Это меню также содержит команду, которая устанавливает режим работы приложения **GIMP**: в одном или нескольких окнах. Также в этом меню находится полный список

диалоговых окон, которые могут постоянно отображаться на экране: набор кистей, цветовые палитры, слои и т. д.

**Help** – система поддержки. В зависимости от того, как было настроено приложение **GIMP**, эта система может быть установлена на вашем локальном компьютере или доступна удаленно. Обратите внимание, что система поддержки доступна не только на английском, но и на румынском и русском языках.

**Приложение GIMP содержит сотни команд, кнопок, курсоров, кассет и других элементов управления. Никакой учебник не мог бы содержать их полное описание. Поэтому узнайте назначение команд и элементов управления, поместив курсор на каждую из них или используя систему поддержки.**

Обычно при активации команды на экране отображается диалоговое окно, в котором пользователь может выбрать желаемые параметры. Например, на *рис. 4.13* показано диалоговое окно **Layers** (Слои), которое отображается на экране сразу после запуска команды **Windows, Dockable Dialogs, Layers** (Окна, Прикрепляемые диалоги, Слои). Это окно содержит стек слоев и элементы управления, которые позволяют пользователю устанавливать их непрозрачность, изменять порядок, в котором они появляются в стеке, исключать их из соответствующего стека, включать новые слои.



*Рис. 4.13.* Основное окно и диалоговые окна **Layers** (Слои) и **Brushes** (Кисти)

На *рис. 4.13* также показано диалоговое окно **Brushes** (Кисти), в котором можно выбирать кисти и устанавливать их параметры.

## 4.9. Практические занятия: Обработка растровых изображений

1. Используя команды в меню приложения **GIMP**, настройте свою рабочую среду:

- установите режим работы в одном или в нескольких окнах;
- определитесь с командами и инструментами, которые вы собираетесь использовать чаще, и настройте способ постоянного отображения соответствующих окон и диалогов на экране;
- приведите в соответствие настройки графического редактора с техническими параметрами компьютера, на котором вы работаете, и с имеющимся цифровым оборудованием: телевизором, видеокамерами, сканером, цветным принтером, мультимедийным проектором.

2. Найдите в меню приложения **GIMP** инструменты, необходимые для выполнения следующих графических операций:

- выделение определенных фрагментов из изображения, подлежащего обработке;
- изменение размера изображения;
- обрезка краев;
- поворот и зеркальное отображение выбранных фрагментов и изображений в целом;
- надпись изображений;
- составление изображений из нескольких слоев;
- изменение цвета;
- изменение яркости и контрастности;
- применение эффектов;
- сжатие изображений.

3. **ЭКСПЕРИМЕНТИРУЙТЕ!** Загрузите 3-4 слоя изображений. Изменяя непрозрачность слоев и их порядок в стеке, обратите внимание, как меняется получившееся изображение.

4. **СОЗДАЙТЕ!** Используя слои и инструменты для работы с ними, создавайте результирующие изображения, представляющие экзотических животных на фоне их среды обитания.

5. **ЭКСПЕРИМЕНТИРУЙТЕ!** Используя видеокамеру, сделайте снимки самых разных объектов. Загрузите эти картинки на свой компьютер. Используя стандартные цветовые каналы, определите, как меняются изображения при их включении или выключении.

6. **ИССЛЕДУЙТЕ!** Используя созданные по вашей инициативе цветовые каналы, определите, как изменение их атрибутов влияет на внешний вид получаемых изображений.

7. **УЧИТЕСЬ УЧИТЬСЯ!** Самый эффективный способ изучить множество эффектов в меню приложения **GIMP** - применить их по отдельности к различным изображениям. Создайте библиотеку изображений, содержащую портреты, пейзажи, спортивные сцены, произведения искусства, архитектурные объекты и т. д., и примените каждый из графических эффектов **GIMP**. Обратите внимание, как обрабатываемые изображения меняются в зависимости от специфики применяемых графических эффектов.

## 4.10. Рекомендуемые проекты

Работая в командах, разработайте один-два из следующих проектов:

1. Цифровые альбомы: жизнь школы, жизнь класса, каникулы, моя деревня / город, портреты, натюрморты, репортажи, путешествия, архитектурные объекты, пейзажи, спортивные соревнования, животные, искусство абстрактной фотографии.

2. Генеалогическое древо одной из известных личностей родом из родного села / города.

3. Коллекции цифровых дидактических фотографий для:

- различных школьных предметов;
- школьного музея, для музея села / родного города;
- личного профиля из Интернета.

4. Цифровые коллекции указателей, знаков (дорожных, охраны труда, предупредительных, информирующих).

5. Тематические выставки цифровых фотографий.

6. Карточки для разных школьных предметов.

7. Тематические выставки цифровой графики.

8. Флаеры, тематические плакаты, афиши:

- права ребенка;
- гражданская активность;
- волонтерство;
- здоровый образ жизни;
- экология и охрана окружающей среды;
- Интернет-безопасность;
- свободная тема.

9. Профили выдающихся личностей.

10. Фотоотчеты с:

- праздничных мероприятий;
- школьных соревнований;
- художественных мероприятий;
- развлекательных мероприятий;
- мероприятий общественной деятельности;
- мероприятий волонтерской деятельности.

11. Художественные тексты:

- замечательные цитаты;
- цитаты с выделением основного смысла;
- девизы.

В процессе разработки проектов вы можете использовать как собственные изображения, снятые с помощью цифровых фотоаппаратов, так и соответствующие изображения, загруженные из Интернета. Просим неукоснительно соблюдать правила безопасности в Интернете, цифровой этики, авторского права.