ИНФОРМАТИКА

В. М. Котов
А. И. Лапо
Ю. А. Быкадоров
Е. Н. Войтехович

8

ИНФОРМАТИКА

Учебное пособие для 8 класса
учреждений общего среднего образования
с русским языком обучения

Допущено
Министерством образования
Республики Беларусь

Минск «Народная асвета» 2018

УДК 004(075.3=161.1)
ББК 32.81я721
И74

Авторы:
В. М. Котов, А. И. Лапо, Ю. А. Быкадоров, Е. Н. Войтехович

Рецензенты:
кафедра современных технологий программирования факультета математики и информатики учреждения образования «Гродненский государственный университет имени Янки Купалы» (кандидат педагогических наук, доцент *Н. П. Макарова*);
учитель информатики высшей квалификационной категории государственного учреждения образования «Средняя школа № 4 г. Дзержинска» *С. Г. Пузиновская*

ISBN 978-985-03-2982-0

СОДЕРЖАНИЕ

Глава 3. Основы алгоритмизации и программирования

Глава 4. Технология обработки текстовых документов

От авторов

Дорогие восьмиклассники! Вы держите в руках учебное пособие по информатике.

Исследования, проводимые в этой науке, востребованы как в естественно-математических, так и в социально-гуманитарных областях знания.

Мы, авторы учебного пособия, постарались сделать так, чтобы оно помогло вам освоить новые знания и углубить уже имеющиеся. Надеемся, что в дальнейшем вы сможете применить полученные умения для решения практических задач из различных предметных областей.

Материал каждого параграфа в данном учебном пособии разделен на две колонки. Цвет фона определяет назначение размещенной на нем информации:

— основной материал, обязательный для изучения;

— примеры, иллюстрирующие основной материал;

— определения основных понятий;

— исторические сведения, информация об ученых, внесших вклад в развитие информатики, и другие интересные факты.

Условные обозначения, которые используются в учебном пособии:

— вопросы для проверки знаний;

— раздел «Упражнения» содержит задания, при выполнении которых используется компьютер;

— раздел «Упражнения» содержит задания для выполнения в тетради;

— раздел «Упражнения» содержит задания, при выполнении которых может быть использована информация, размещенная в электронном образовательном ресурсе на Национальном образовательном портале (http://e-vedy.adu.by). В таких заданиях вам будет предложено открыть или загрузить файл. Кроме ссылки, вы можете воспользоваться матричным QR-кодом:

* — задание или пример для любознательных.

Имя файла для скачивания содержит номер параграфа и номер упражнения после этого параграфа. Например, имя файла upr3_1 означает, что файл относится к первому упражнению после третьего параграфа. Также на портале размещены файлы с программами, рассмотренными в примерах. Такие файлы имеют имя Program13_5.pas (программа для примера 13.5).

В этом учебном году вы сможете научиться обрабатывать звук и видео, создавать анимированные фильмы. Также вы продолжите изучать алгоритмы и программирование и усовершенствуете свои навыки по созданию и оформлению текстового документа. Вам предстоит освоить новые инструменты в известных программах и познакомиться с новыми программами. В учебном пособии много различного иллюстративного материала. Экранные копии предназначены для первоначального ознакомления с интерфейсами программ, для указания расположения отдельных элементов. Подробно рассмотреть все структурные элементы окна используемой программы можно на экране компьютера.

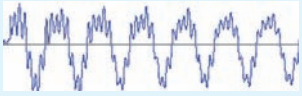
К данному учебному пособию разработано электронное приложение, размещенное на Национальном образовательном портале.

Желаем вам успехов в изучении информатики!

Глава 1
ТЕХНОЛОГИЯ ОБРАБОТКИ АУДИО- И ВИДЕОИНФОРМАЦИИ

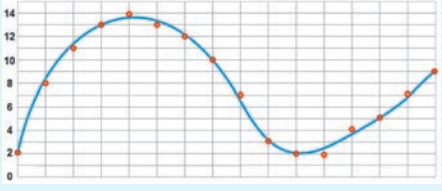
§ 1. Запись аудио- и видеоинформации

Пример 1.1. Аналоговая форма звукового сигнала.



История записи аналогового звука началась в 1857 г., когда француз Эдуард Леон Скотт де Мартенвиль изобрел прибор, выцарапывавший звуковые дорожки на закопченной бумаге.

Пример 1.2. Схема оцифровки аналогового звукового сигнала.



Волна заменяется набором точечных (импульсных) сигналов, а величина импульсов задается числами. Набор точек преобразуется в набор чисел: **(2)(8)(11)(13)(14)(13)(12)(10)(7)...**

История кинематографа началась в 1885 г., когда французы Огюст и Луи Люмьеры впервые провели демонстрацию кинофильмов.

Пример 1.3. Набор кадров цифрового видео можно воспринимать как набор электронных кадров кинофильма.

1.1. Аудио- и видеофайлы

Аудиоинформация (звукозапись) — звуковая информация, записанная каким-либо образом, пригодным для воспроизведения.

Звуковые колебания воздуха (звуковые сигналы) имеют форму, которую называют **аналоговой** (пример 1.1).

Ранее аудиоинформация в аналоговой форме записывалась в студиях, а воспроизводилась с помощью фонографов, граммофонов, патефонов, магнитофонов и электропроигрывателей.

С началом компьютерной эры звуковые сигналы начали оцифровывать, т. е. волны стали заменять наборами точечных (импульсных) сигналов, а величину импульсов — числовыми кодами (пример 1.2). Аудиоинформация получила **цифровую** форму.

Аудиофайл — файл с аудиоинформацией в цифровой форме.

Видеоинформация — изображение движущихся объектов, записанное каким-либо образом, пригодным для воспроизведения.

Сначала видеоинформацию записывали **в форме кинофильмов**. При демонстрации отдельные фотокадры на кинопленке сливались на экране в движущееся изображение.

Видеоинформация **в цифровой форме** является набором электронных фотографий (пример 1.3).

Видеофайл — файл с видеоинформацией и сопровождающей ее аудиоинформацией в цифровой форме.

1.2. Программные средства записи и воспроизведения

В смартфонах программные средства записи звука при помощи микрофона представлены диктофонами (пример 1.4).

На компьютерах с операционной системой Windows стандартной программой для записи звука при помощи микрофона является программа «Звукозапись» (пример 1.5). Более широкие возможности имеет бесплатная программа UV SoundRecorder.

Для записи видеоинформации в смартфонах широко используется приложение «Камера» в режиме записи видео. Этот режим имеют и цифровые фотоаппараты.

Записывать видео позволяют компьютеры с микрофоном и веб-камерой. Для компьютеров разработаны также программы для записи звука и видео, воспроизводимых другими программами.

Программы для воспроизведения аудио- и видеофайлов называются **плеерами**. **Медиаплееры** — плееры, которые воспроизводят как звук, так и видео (примеры 1.6 и 1.7).

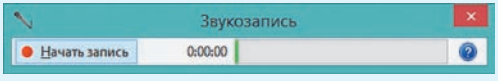
Аудио- и видеофайлы могут иметь лицензионные ограничения на бесплатное копирование, воспроизведение и распространение.

1.3. Форматы аудиофайлов

Аудиофайлы, как и видеофайлы, могут различаться способами цифровой записи — форматами.

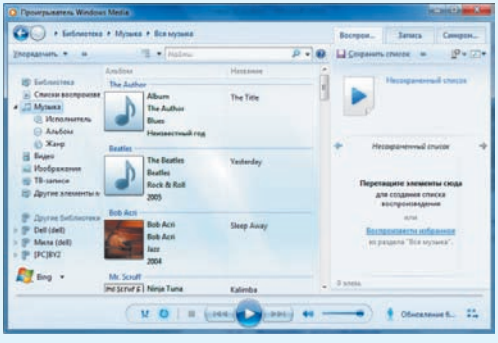
Пример 1.4. Приложение «Диктофон» в одном из современных смартфонов.

Пример 1.5. Окно стандартной для Windows программы «Звукозапись».



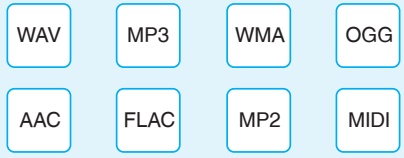
Пример 1.6. На компьютерах чаще всего устанавливаются аудиоплееры AIMP, Winamp Lite, медиаплееры Windows Media Player (WMP), Winamp, Quicktime, KM Player, VLC Media Player и др. К настоящему времени разработан целый ряд аудио- и медиаплееров для смартфонов.

Пример 1.7. Окно медиаплеера WMP.



Кнопки управления воспроизведением размещены в нижней части окна.

Пример 1.8. Названия основных форматов аудиофайлов:



Цифровая звукозапись может иметь несколько каналов: моно (1 канал), стерео (2 канала), Dolby Digital (6 каналов) и т. д.

Пример 1.9. Имена аудиофайлов разных форматов:

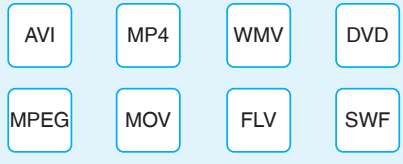
`golos2.wav;` `karaoke.midi;`
`Скрипка.mp3;` `test31.wma.`

Пример 1.10. Одна минута записи в формате WAV имеет объем около 10 Мб, в формате MP3 — от 0,5 до 2,4 Мб.

Формат WMA (Windows Media Audio) разработан компанией Microsoft как конкурент формата MP3 и включает поддержку системы управления авторскими правами.

Это означает, что прослушивать защищенные композиции можно только на компьютере, с которого композиция была загружена из музыкального магазина.

Пример 1.11. Наиболее популярные форматы видеофайлов:



Компания Apple активно использует собственные обозначения для форматов видеофайлов, аналогов MP4, например `m4a`, `m4b`, `m4v`, `m4p`, `m4r`.

Для мобильных телефонов разработан формат 3GP, который использует мощное сжатие. Это позволяет использовать его на слабых мобильных телефонах.

Формат аудиофайла — структура и особенности записи в файле цифровой аудиоинформации.

Существует более 40 форматов аудиофайлов (пример 1.8). Название формата служит расширением имени (типом) аудиофайла (пример 1.9).

Для высококачественной записи звука используется формат WAV. По сравнению с файлами других форматов файлы этого формата имеют очень большие объемы.

Формат MP3 самый распространенный. Использует специальные методы сжатия аудиофайлов за счет небольшого снижения качества звука.

(Рассмотрите пример 1.10.)

Формат MIDI (MID) является цифровым представлением нотных записей для использования на электронных музыкальных инструментах. Воспроизведению разных нотных партий можно придать окраску звуков фортепиано, скрипки, трубы и других инструментов.

1.4. Форматы видеофайлов

Формат видеофайла — структура и особенности записи в файле цифровой видеоинформации и сопровождающей ее аудиоинформации.

Форматы для записи в видеофайл только видеоинформации не предусмотрены. Но видео можно сохранять в файле и без звука.

Известно более 70 форматов видеофайлов (пример 1.11). Название формата служит расширением имени (типом) видеофайла. Для записи аудио- и видеофайлов, кроме программ записи, используют кодеки.

Кодек — специальная программа, которая сжимает (уменьшает) и восстанавливает первоначальный объем аудио- или видеофайла.

Различают **аудиокодеки** и **видеокодеки.** Аудиокодеки носят имена форматов аудиофайлов. Имена видеокодеков с именами форматов не совпадают (пример 1.12).

При записи и воспроизведении видеофайла всегда используется пара из видеокодека и аудиокодека. Форматы видеофайлов используют разные пары кодеков (пример 1.13).

Пример 1.12. Названия основных видеокодеков:

H.261 | XviD | DivX | MPEG4

Пример 1.13. Популярный формат видеофайлов AVI может использовать видеокодек H.264 и аудиокодек MP3.

Возможны и другие сочетания, например видеокодек MPEG4 и аудиокодек AC3, видеокодек XviD и аудиокодек MP3.

1. Что такое аудиоинформация?
2. В каком виде записывается аудиоинформация в цифровой форме?
3. Что такое видеоинформация?
4. Что такое видеоинформация в цифровой форме?
5. Как называются программы воспроизведения цифровых аудио- и видеофайлов?
6. Что такое формат аудиофайла?
7. Чем интересен формат аудиофайлов MIDI?
8. Что такое формат видеофайла?
9. Существуют ли форматы для записи в видеофайл только видеоинформации?
10. Что такое кодек?
11. Какие виды кодеков используются для работы с видеофайлами?
12. Сколько кодеков требуется для записи видеофайла?

Упражнения

1 Приведите примеры форматов аудиофайлов.

2 Приведите примеры форматов видеофайлов.

3 Приведите примеры аудиокодеков.

4 Приведите примеры видеокодеков.

5 Откройте в смартфоне приложение «Диктофон». Произнесите и запишите определение формата видеофайла. Воспроизведите запись.

6 С помощью соответствующего приложения найдите в смартфоне папку с аудиофайлами и определите их форматы.

7 Откройте в смартфоне приложение «Камера». Попросите одноклассника прочитать вслух определение формата аудиофайла перед камерой. Запишите видео и воспроизведите его.

8 С помощью подходящего приложения найдите в смартфоне папку с видеофайлами и определите их форматы.

9 Включите компьютер, подключите к нему микрофон и наушники. Произнесите определение формата видеофайла и с помощью программы «Звукозапись» запишите его в аудиофайл.

§ 2. Введение в редактирование аудиофайлов

Пример 2.1. Необходимость в редактировании звукозаписи возникает, когда ее длительность нужно уменьшить или увеличить. Например, когда длительности подобранной музыкальной композиции недостаточно для сопровождения готовящегося выступления певцов или танцоров. В таких случаях какой-то фрагмент звукозаписи дублируют несколько раз.

Применение звукового эффекта позволяет изменить стиль звучания звукозаписи, например громкость звучания, скорость или темп воспроизведения, высоту тона. С помощью звуковых эффектов можно удалять щелчки и треск, добавлять эхо, удалять из музыкальной композиции звучание голоса.

Пример 2.2. Среди бесплатных аудиоредакторов выделим Audacity, WavePad Sound Editor, Wavosaur, FREE Wave MP3 Editor, Swifturn Free Audio Editor.

Пример 2.3. Конвертация аудиофайла может понадобиться, например, если в мультимедийную презентацию нужно вставить звуковую запись с CD-диска. Программа для создания мультимедийных презентаций не допускает вставку на слайд аудиофайлов такого формата.

Пример 2.4. Аудиоредакторы позволяют сохранять аудиофайлы в разных форматах, поэтому для конвертации достаточно загрузить аудиофайл в одном формате, а потом сохранить его в другом.

2.1. Редактирование и конвертация

Известны два вида обработки аудиофайлов: редактирование и конвертация.

Редактирование аудиофайла — процесс его изменения, который заключается в вырезании, вставке, удалении и комбинировании частей аудиофайла, которые называются **фрагментами.** Редактирование включает также применение звуковых эффектов ко всей звукозаписи и к ее фрагментам (пример 2.1).

Для редактирования аудиофайлов используются программные средства, которые называются **аудиоредакторами** (пример 2.2).

Редактировать аудиофайлы мы будем с помощью аудиоредактора Audacity[1]. С интерфейсом данной программы можно познакомиться в *Приложении 1* (с. 153).

Конвертация аудиофайла — процесс изменения его формата (пример 2.3). Чтобы выполнить конвертацию аудиофайлов, можно использовать аудиоредакторы (пример 2.4).

2.2. Загрузка и воспроизведение звукозаписи в аудиоредакторе

Загрузку аудиофайла в редактор Audacity начинают командой главного меню **Файл** → **Открыть ...** .

[1] Доступен для скачивания на сайте https://www.audacityteam.org

Аудиоредактор автоматически конвертирует файл в свой внутренний формат, и в окне появляется изображение звукозаписи в виде одной или двух аудиодорожек (треков).

Воспроизводить звукозапись, приостанавливать и останавливать воспроизведение позволяют первые три кнопки **Панели воспроизведения и записи** (пример 2.5).

Во время воспроизведения вправо по дорожкам движется вертикальная линия — **курсор**. Воспроизведение звукозаписи всегда начинается с заданного положения курсора. Когда воспроизведение остановлено, курсор можно перенести в любое другое место дорожки (пример 2.6).

2.3. Выделение фрагмента звукозаписи

В аудиоредакторе любой фрагмент звукозаписи можно выделить. Заметим, что на **Панели инструментов** редактора должна быть нажата кнопка **Выделение** (пример 2.7).

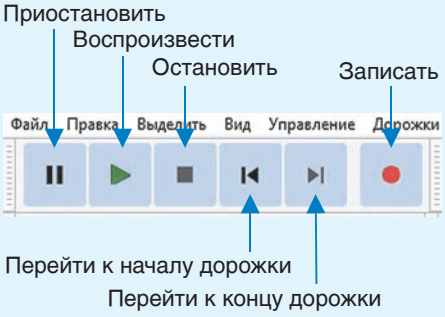
Выделенный фрагмент на дорожке получает другой цвет фона (пример 2.8). Если фрагмент звукозаписи выделен, то воспроизвести и прослушать можно только его. Выделение фрагмента снимается щелчком мыши по свободному месту дорожки.

Различают два способа выделения фрагментов: обзорный и точный.

Обзорный способ выделения фрагментов используется с целью прослушивания фрагментов и проводится протяжкой указателя мыши по дорожке.

Границы выделенного фрагмента всегда можно переместить. Для этого

Пример 2.5. Кнопки **Панели воспроизведения и записи.**

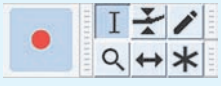


Названия кнопок отражают их функции.

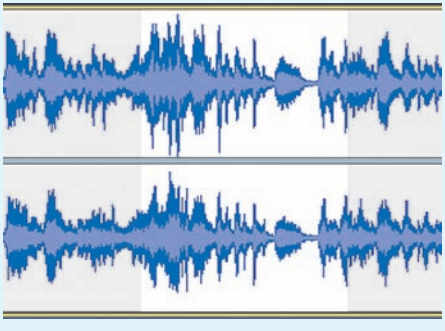
Воспроизведение звукозаписи также можно начать/остановить нажатием клавиши **Пробел** на клавиатуре.

Пример 2.6. Курсор можно перенести в другое место дорожки щелчком мыши. Точно в начало дорожки курсор переносится кнопкой **Перейти к началу** дорожки панели **Воспроизведения и записи**, а точно в конец дорожки — кнопкой **Перейти к концу дорожки.**

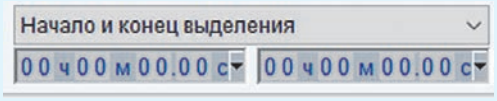
Пример 2.7. Кнопка **Выделение** на **Панели инструментов.**



Пример 2.8. Изображение выделенного фрагмента звукозаписи.



Пример 2.9. Числовые поля на **Панели выделения фрагментов.**



Поле **Начало выделения** показывает отсчет начала выделенного фрагмента.

Поле **Конец выделения** показывает отсчет конца выделенного фрагмента.

Поле **Позиция аудио** показывает отсчет положения курсора.

Пример 2.10. Введем в числовое поле **Начало выделения** отсчет 6,9 с:

1. Щелкаем мышью по разряду секунд в поле (разряд выделяется белым цветом).

2. Колесиком мыши устанавливаем значение 6.

3. Аналогично после щелчка в разряд десятых долей секунды вводим цифру 9.

Пример 2.11. Кнопка **Обрезать** на **Панели редактирования.**

Пример 2.12. Обрежем звукозапись до длительности 2 мин 30 с.

Точным способом выделяем фрагмент длительностью 2 мин 30 с от начала записи и нажимаем кнопку на **Панели редактирования.**

Пример 2.13. Для применения эффекта **Плавное затухание** в конце записи выделяют фрагмент длительностью 2—4 с и используют команду **Эффекты.** Выпадает большой список эффектов, в котором нужно найти требуемый.

достаточно подвести указатель мыши к границе выделенного фрагмента изнутри (указатель принимает вид) и перетащить границу.

Точный способ выделения фрагментов используется для их копирования. Этим способом фрагмент выделяется с помощью отсчетов времени от начала записи на **Панели выделения фрагментов** (пример 2.9).

Чтобы выделить фрагмент, в поля **Начало и Конец выделения** вводят время начала и конца фрагмента соответственно (пример 2.10).

2.4. Обрезка фрагмента звукозаписи и применение эффекта

Операция обрезки используется в случаях, когда длительность звукозаписи нужно сократить. В звукозаписи выделяется фрагмент нужной длительности и используется кнопка **Обрезать** на **Панели редактирования** (пример 2.11). В итоге в звукозаписи остается только выделенный фрагмент (пример 2.12).

Чтобы звук обрезанной звукозаписи при воспроизведении не обрывался резко, нужно применить один из эффектов, например **Плавное затухание** (эффект плавного уменьшения громкости звучания) (пример 2.13).

2.5. Сохранение аудиофайла

Для сохранения аудиофайла в редакторе Audacity есть две возможности.

Если работу со звукозаписью надо продолжить, то командой **Файл** → **Сохранить проект** звукозапись сохраняют во внутреннем формате редактора

как файл проекта с расширением **.aup** (тип **AUP**). Воспроизвести такой аудиофайл на плеерах невозможно.

Чтобы сохранить аудиофайл в другом формате, его экспортируют (конвертируют). Для этого командой **Файл** → → **Export Audio...** вызывается окно **Export Audio**, в котором вводится имя аудиофайла и выбирается его формат (пример 2.14).

Кнопка **Параметры** в окне **Export Audio** позволяет вызвать окно для установки параметров качества сохраняемой звукозаписи. Основной параметр качества цифровой записи звука и видео носит название *битрейт*.

Битрейт (скорость потока) — количество бит двоичной записи, которое приходится на секунду воспроизведения.

Битрейт измеряется в килобитах в секунду (кбит/с, или kbps).

Чем больше битрейт, тем выше качество записи и больше объем файла (пример 2.15).

Пример 2.14. Форматы, которые поддерживает редактор Audacity при загрузке и экспорте аудиофайлов.

AIFF (Apple) signed 16 bit PCM
WAV (Microsoft) signed 16 bit PCM
GSM 6.10 WAV (mobile)
Файлы MP3
Файлы Ogg Vorbis
Файлы FLAC
Файлы MP2
Передать внешней программе
M4A (AAC) Files (FFmpeg)
AC3 Files (FFmpeg)
AMR (narrow band) Files (FFmpeg)
WMA (version 2) Files (FFmpeg)
Custom FFmpeg Export

Пример 2.15. Связь величины битрейта и качества двухканальной звукозаписи в формате MP3:

- **32 кбит/с** — качество записи речи в диктофонах;
- **96 кбит/с** — качество записи для передачи речи или звука низкого качества по каналам связи;
- **192 кбит/с** — приемлемый уровень качества для записи музыки;
- **256 кбит/с** — высокий уровень качества для записи музыки;
- **320 кбит/с** — наивысший уровень качества звукозаписи, поддерживаемый форматом MP3.

1. Что такое курсор аудиоредактора Audacity?
2. Каким образом выделенный фрагмент звукозаписи отображается в редакторе Audacity?
3. Чем различаются обзорный и точный способы выделения фрагментов?
4. Как производится обрезка фрагмента звукозаписи?
5. Какое изменение фрагмента производит эффект **Плавное затухание**?
6. Какие возможности для сохранения аудиофайлов имеет редактор Audacity?
7. Что такое битрейт?

Упражнения

1 Откройте в аудиоредакторе файл с музыкальной композицией (данная композиция лицензионных ограничений не имеет). Прослушайте ее.

2 Выделите любой фрагмент загруженной звукозаписи обзорным способом. Воспроизведите его с помощью кнопок **Панели воспроизведения и записи** и с помощью клавиши **Пробел**.

3 Установите, чем различаются действия кнопок **Остановить** и **Приостановить**?

4 Выделите и прослушайте фрагмент загруженной звукозаписи от 31 с до 1 мин 27 с.

5 Сократите звучание загруженной звукозаписи до 1 мин с использованием эффекта **Плавное затухание**.

6 Сохраните результат предыдущего упражнения как проект формата AUP и как аудиофайл формата MP3 с битрейтом 192 кбит/с.

§ 3. Основные операции редактирования аудиофайла

Основные задачи редактирования аудиофайла удобно рассмотреть на примере редактирования музыкальных композиций. Задача сокращения музыкальной композиции больших трудностей не вызывает. Композицию достаточно обрезать и в конце приглушить звук.

Структура музыкальной композиции определяется набором ее фрагментов. Многие музыкальные композиции (песни) имеют форму, которую называют куплетной, т. е. состоящей из куплетов. В куплете, как правило, два фрагмента: запев и припев. В композиции также возможны фрагменты, которые называют вступлениями и проигрышами.

Пример 3.1. Музыкальные композиции, используемые для сопровождения выступлений на концертах или капустниках (фонограммы), часто требуют увеличения или уменьшения числа куплетов. Это связано с тем, что самодельные тексты песен редко совпадают по числу куплетов с исходными фонограммами.

Кроме того, в таких фонограммах бывает необходимо удалить или поменять местами припевы или проигрыши.

При создании фонограмм для видеофильмов длительности одной музыкальной композиции часто также не хватает на весь фильм. В таком случае ее нужно увеличивать дублированием всей композиции или некоторых ее фрагментов.

3.1. Основные задачи редактирования

Основными задачами редактирования аудиофайла являются:

- сокращение аудиофайла;
- изменение структуры фрагментов аудиофайла.

Сокращение аудиофайла требуется, когда длительность его звучания превышает нужную, например требуемую длительность звучания музыкального сопровождения. Один из способов решения этой задачи рассмотрен в предыдущем параграфе.

Задача изменения структуры фрагментов аудиофайла возникает, когда отдельные фрагменты звукозаписи нужно удалить, переставить или продублировать (пример 3.1).

Для музыкальных композиций задача усложняется тем, что выделять фрагменты (куплеты, припевы и т. д.) следует точным способом. Чтобы не нарушить плавности звучания, отсчеты времени для начала и конца фрагментов определяют максимально точно.

3.2. Алгоритм нахождения точного отсчета

Аудиоредактор Audacity позволяет с высокой точностью находить отсчеты для моментов начала куплетов, припевов и других фрагментов с использова-

нием инструментов масштабирования. Инструменты масштабирования в аудиоредакторе применяются для увеличения или уменьшения изображения звукозаписи на дорожке (пример 3.2).

Для нахождения точного отсчета для моментов начала музыкальных фрагментов будем использовать следующий алгоритм:

1. Прослушать композицию и с любой точностью выделить фрагмент, который содержит нужный момент.

2. Щелкнуть по кнопке **Уместить выделение**. Изображение фрагмента увеличивается, но выделение не снимается.

3. Прослушать выделенный фрагмент (можно несколько раз). В нужный момент, определяемый на слух, щелкнуть по кнопке **Приостановить**. Курсор замирает. Требуемый момент определен, а отсчет времени показан в поле **Текущая позиция**.

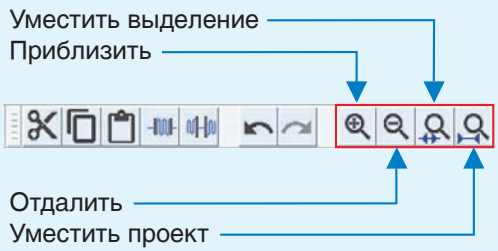
4. Для уточнения отсчета около положения курсора выделить небольшой фрагмент и щелкнуть по кнопке **Остановить**. Курсор перемещается в начало выделенного фрагмента. Далее перейти к шагу 2.

Уточнение достаточно провести 2 раза (пример 3.3).

3.3. Основные операции редактирования

Редактирование аудиофайлов включает следующие основные операции с фрагментами: **выделение, обрезку, копирование, вставку, удаление** и **применение эффекта**.

Пример 3.2. Группа кнопок инструментов масштабирования на **Панели редактирования**:



1. Кнопка **Приблизить** увеличивает изображение звукозаписи.

2. Кнопка **Отдалить** уменьшает изображение звукозаписи.

3. Кнопка **Уместить выделение** показывает весь выделенный фрагмент.

4. Кнопка **Уместить проект** показывает всю звукозапись.

Пример 3.3. Определим отсчет времени для начала второго куплета в некоторой музыкальной композиции, исполняя шаги алгоритма:

1. Композиция прослушивается, и выделяется фрагмент с моментом начала второго куплета.

2. Кнопкой **Уместить выделение** изображение фрагмента увеличивается.

3. Выделенный фрагмент прослушивается, и в нужный момент проводится щелчок по кнопке **Приостановить**.

4. Около положения курсора выделяется небольшой фрагмент и проводится щелчок по кнопке **Остановить**.

5. Повторяются шаги 2—4.

6. Повторяются шаги 2—3.

Отсчет — в поле **Текущая позиция**.

Пример 3.4. Кнопки удаления, копирования и вставки фрагментов на **Панели редактирования**.

Пример 3.5. Увеличим на один куплет длительность музыкального сопровождения к песне.

В соответствии с алгоритмом из пункта 3.2 устанавливаем отсчеты для моментов начала и конца второго куплета.

По отсчетам точным способом выделяем фрагмент записи для копирования. Фрагмент копируем в буфер обмена с помощью кнопки **Копировать** и нажимаем клавишу-стрелку **Вправо** клавиатуры. В результате выделение пропадает, а курсор смещается в конец использованного фрагмента.

Щелкаем по кнопке **Вставить**. Фрагмент (куплет) вставляется.

При необходимости операцию вставки можно повторить несколько раз.

С операциями выделения, обрезки и применения эффекта мы уже познакомились. Операции копирования, вставки и удаления фрагментов известны вам по другим программам и проводятся при помощи кнопок на **Панели редактирования** (пример 3.4). Вместо этих кнопок можно использовать комбинации клавиш клавиатуры, известные вам из курса 6-го класса.

Операции копирования, вставки и удаления фрагментов используются для изменения структуры фрагментов аудиофайла (пример 3.5).

Операции редактирования при необходимости можно отменить и вернуть, используя кнопки **Отменить** и **Вернуть** на **Панели редактирования**. Эти кнопки позволяют отменить или вернуть целый набор последних операций редактирования.

1. Какие задачи являются основными при редактировании аудиофайла?
2. Почему при редактировании моменты начала куплета или припева в песне нужно находить с максимальной точностью?
3. Для чего предназначены инструменты масштабирования в аудиоредакторе?
4. Какие операции редактирования аудиофайла являются основными?
5. Какие операции редактирования аудиофайла обеспечивают изменение структуры аудиофайла?
6. Какие кнопки на **Панели редактирования** используются при копировании, вставке и удалении фрагментов?

Упражнения

1 Откройте в аудиоредакторе файл с музыкальной композицией (данная композиция лицензионных ограничений не имеет). Прослушайте ее.

2 Выделите и прослушайте фрагмент загруженной звукозаписи от 2 мин 08 с до 2 мин 40 с.

3 Увеличьте изображение фрагмента, выделенного в задании 2, на всю дорожку. Верните изображение всей звукозаписи.

4 Загруженная композиция начинается с припева, включает два куплета и заканчивается половиной припева. Удалите второй куплет и половину припева после него. На это место вставьте два раза первый куплет.

5 Сохраните результат упр. 4 как проект формата AUP и как аудиофайл формата MP3 с максимальным битрейтом. Прослушайте полученную композицию.

§ 4. Введение в компьютерный видеомонтаж

4.1. Видеомонтаж и конвертация

Все видеофайлы делятся на **видеофрагменты**, которые получены в результате записи, и **видеофильмы**, которые предназначены для демонстрации.

Для видеофайлов известны два вида обработки: видеомонтаж и конвертация.

Компьютерный видеомонтаж — это процесс создания видеофильма из видеофрагментов с помощью специального программного средства.

Программное средство для видеомонтажа называется **видеоредактором** (пример 4.1). Мы будем использовать видеоредактор VideoPad[1]. С интерфейсом программы можно познакомиться в *Приложении 1* (с. 154).

Конвертация видеофайла заключается в изменении его формата (пример 4.2). Для конвертации видеофайлов используются программные средства, которые называются **видеоконвертерами** (пример 4.3). Мы будем использовать видеоконвертер Convertilla[2]. С интерфейсом программы можно познакомиться в *Приложении 1* (с. 155).

Пример 4.1. Среди бесплатных видеоредакторов для компьютеров выделим редакторы Videopad, Shotcut, OpenShot Video Editor, Windows Movie Maker (или **Киностудия**), Lightworks.

Разработаны видеоредакторы и для смартфонов на Andriod: Video Editor, PowerDirector, KineMaster — Pro Video Editor. Есть они на других платформах.

Пример 4.2. На современных компьютерах установлены кодеки для известных форматов видеофайлов. О портативных устройствах такого сказать нельзя. Поэтому видео, закачанное на смартфон, часто не открывается, и его обычно подвергают конвертации. Для видеомонтажа иногда приходится конвертировать видеофрагменты, снятые на разных мобильных устройствах.

Пример 4.3. Многие видеоконвертеры позволяют конвертировать и аудиофайлы. Среди бесплатных видеоконвертеров для компьютеров следует выделить Convertilla, VSDC Free Video Converter, Any Video Converter Free, Format Factory и др.

Конвертировать видеофайлы можно и с помощью онлайн-видеоконвертеров[3].

---

[1] Доступен для скачивания на сайте https://www.nchsoftware.com/videopad/ru

[2] Доступен для скачивания на сайте http://convertilla.com/ru/

[3] Информация получена с сайтов https://convert-video-online.com/ru/ и https://www.online-convert.com/ru (дата доступа 10.01.2018).

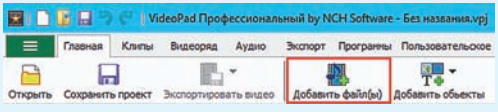
Пример 4.4. Одно из возможных визуальных представлений видеоряда.

Создавая видеофильм из фрагментов, видеоряд можно изменять — удалять, добавлять и перемещать видеофрагменты.

Пример 4.5. Текстовые видеофрагменты позволяют включать в видеофильм текстовые надписи (название фильма, его частей и др.).

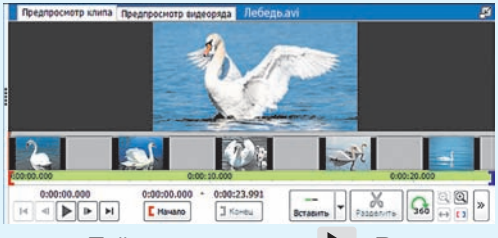
Пример 4.6. Все видеофрагменты, как правило, имеют собственное звуковое сопровождение. В результате нарезки видеофрагментов и сборки фильма этот вид звукового сопровождения становится фрагментарным и очень плохо воспринимается. Требуется добавление фонограммы.

Пример 4.7. Кнопка **Добавить файл(ы)** на вкладке **Главная**.



Загруженные файлы автоматически распределяются по разделам (папкам) **Видеофайлы** и **Аудиофайлы**. В разделе **Видеоряды** находится автоматически создаваемый файл проекта.

Пример 4.8. Видеоклип в окне предпросмотра.



Действие кнопки ▶ **Воспроизведение/Пауза** дублирует клавиша **Пробел**. Видеоклип можно воспроизводить покадрово.

4.2. Основные операции видеомонтажа

Компьютерный видеомонтаж включает следующие основные операции:

- деление и обрезку видеофрагментов;
- создание видеофильма из фрагментов;
- сохранение видеофильма.

Создание видеофильма из фрагментов использует понятие видеоряда.

Видеоряд — полоса из условных изображений видеофрагментов, которая отражает структуру видеофильма (пример 4.4).

При необходимости компьютерный видеомонтаж может также включать следующие операции:

- создание и добавление в видеофильм текстовых видеофрагментов (пример 4.5);
- добавление в видеофильм музыкального сопровождения (фонограммы) из внешнего аудиофайла (пример 4.6).

4.3. Загрузка, деление и обрезка видеофрагментов

Видеоредактор VideoPad позволяет загружать как видеофайлы (видеофрагменты), так и аудиофайлы (фонограммы). Загрузку проводят с помощью кнопки **Добавить файл(ы)** на вкладке **Главная** (пример 4.7).

Загруженные файлы в видеоредакторе VideoPad называются **клипами**.

Если открыты вкладки **Главная** или **Клипы**, то щелчок по видеоклипу автоматически открывает его в окне предпросмотра на вкладке **Предпросмотр клипа** (пример 4.8). В этом окне видеофрагмент можно просмотреть, разделить или обрезать.

Курсор в окне предпросмотра — вертикальная красная линия на полосе эскизов и шкале времени, которая показывает положение текущего кадра в клипе.

Видеоклип можно разделить на две части по положению курсора (кнопкой **Разделить**).

Для обрезки клипа достаточно перетащить в новое положение по шкале времени красную и синюю скобки (пример 4.9).

Клип обрезается виртуально. Это означает, что границы обрезки всегда можно сдвинуть.

4.4. Создание видеофильма из фрагментов

Создание видеофильма заключается в сборке видеоряда из видеофрагментов.

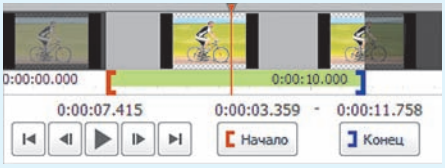
Клипы из раздела **Видеофайлы** по одному переносятся в окно видеоряда, которое должно находиться в режиме **Раскадровка**. Клип можно перетащить мышью либо выделить его и использовать комбинацию клавиш: **Ctrl** + + **Shift** + **End** — в конец видеоряда, **Ctrl** + **Shift** + **Home** — в начало видеоряда (пример 4.10).

В окне предпросмотра теперь можно просмотреть и видеоряд. Переключение между объектами просмотра проводят выбором вкладок в заголовке окна предпросмотра (пример 4.11).

4.5. Сохранение видеофильма

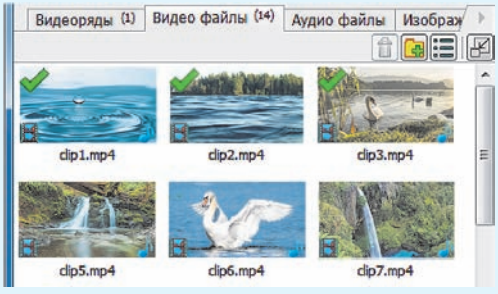
Сохранение видеофильма как проекта в файле с расширением **.vpj** внутреннего формата начинается кнопкой **Сохранить проект** на вкладке **Главная**.

Пример 4.9. Обрезка видеоклипа в окне предпросмотра.



Числовые поля над кнопками **Начало** и **Конец** открываются для ввода щелчком мыши. Отсчеты времени начала и конца оставляемой части клипа можно ввести точно в числовых полях над кнопками **Начало** и **Конец**.

Пример 4.10. Видеофайлы, которые перенесены в видеоряд, в разделе **Видеофайлы** автоматически отмечаются зеленой «галочкой».



Пример 4.11. Вкладка **Предпросмотр видеоряда** окна предпросмотра имеет свою систему кнопок управления.

В набор кнопок управления воспроизведением добавляются две кнопки: **Перейти к краю предыдущего клипа** и **Перейти к краю следующего клипа**.

Кнопка **Разделить** получает меню, в котором можно выбрать дорожки для разделения. Появляется кнопка **Снимок**, которая позволяет сделать снимок кадра видеофильма. Кнопка **Включить 360 гр** предназначена для добавления сложного видеоэффекта.

Курсор видеоряда находится в окне видеоряда, а управлять им можно как мышью в окне видеоряда, так и кнопками в окне предпросмотра.

Пример 4.12. Окно **Экспортировать видео** в разделе **Настройки экспорта файла** содержит 7 настраиваемых параметров.

В поле 1-й строки **Имя файла:** вводится имя создаваемого видеофайла.

В поле 2-й строки **Сохранить в:** вводится имя папки для сохранения файла. Имя папки выбирается в окне, которое вызывается в той же строке кнопкой **Обзор**.

В поле 4-й строки **Формат:** название формата будущего видеофайла выбирается в списке форматов. При необходимости можно использовать **Продвинутые настройки кодека**.

Остальные настройки экспорта рекомендуется не изменять.

В окне **Сохранить проект как** выбирается папка и имя проекта.

Сохранение видеофильма в другом формате называется *экспортом*.

Экспорт начинается щелчком по кнопке **Видеофайл** на вкладке **Экспорт**. В ответ открывается окно **Экспортировать видео**, где проводится настройка параметров экспорта видеофайла (пример 4.12).

После настройки параметров щелчок по кнопке **Создать** в этом окне начинает довольно длительный процесс сохранения видеофильма, ход которого отображается в другом диалоговом окне **Очередь на экспорт**.

1. Что такое компьютерный видеомонтаж?
2. Какие основные операции включает компьютерный видеомонтаж?
3. Какие операции, кроме основных, может включать компьютерный видеомонтаж?
4. Как проводят загрузку видео- и аудиофайлов в редактор VideoPad?
5. Для чего в редакторе VideoPad предназначено окно предпросмотра?
6. Что такое курсор в окне предпросмотра?
7. Как проводится обрезка видеоклипов в редакторе VideoPad?
8. Можно ли изменить обрезку видеоклипа в редакторе VideoPad?
9. В каком режиме окна видеоряда проводится создание видеоряда?
10. Каким образом в редакторе VideoPad клипы перемещаются в видеоряд?
11. Какой элемент окна предпросмотра отвечает за выбор между просмотром клипа и видеоряда?
12. Что такое сохранение проекта в редакторе VideoPad?
13. Что такое экспорт видеофильма в редакторе VideoPad?

Упражнения

1 В видеоредакторе VideoPad создайте новый проект и откройте (загрузите в видеоредактор) готовые видеофрагменты (все видеофрагменты лицензионных ограничений не имеют).

2 Выполните перечисленные действия, используя редактор VideoPad.

1. Обрежьте все видеозаписи из раздела **Видеофайлы** до длительности 4 с.
2. Создайте из фрагментов видеоряд в порядке их нумерации.
3. Просмотрите полученный видеоряд.
4. Сохраните созданный видеофильм как проект в файле с именем **Вода.vpj**.
5. Сохраните созданный видеофильм в видеофайле **Вода.avi**.

3 Используя конвертер Convertilla, сконвертируйте файл **Вода.avi** в формат **MP4**.

Операция	Результат
Запустить программу Convertilla.	Появляется окно программы.
Щелкнуть по кнопке **Открыть**.	Появляется окно **Выбор файла видео**.
В окне **Выбор файла видео** найти и выделить файл **Вода.avi**, а затем щелкнуть по кнопке **Открыть**.	В нижнем поле **Файл:** предлагаются папка и имя файла для сохранения результата конвертации, которые можно изменить.
В списке **Формат:** выбрать MP4.	Кодеки подбираются автоматически.
Щелкнуть по кнопке **Конвертировать**.	Видеофайл конвертируется и сохраняется в формате MP4.

§ 5. Компьютерный видеомонтаж с текстами и фонограммой

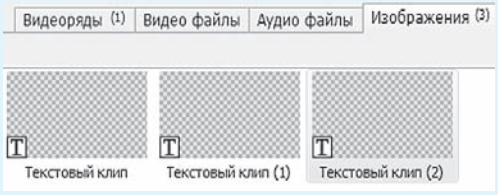
5.1. Создание текстовых клипов

Текстовый клип — видеофрагмент с текстовой надписью на прозрачном или цветном фоне. Для создания текстового клипа на вкладке **Клипы** используют кнопку **Добавить текст**. Новый клип под именем **Текстовый клип** появляется в разделе **Изображения** для исходных файлов. Для последующих текстовых клипов к имени добавляется номер в скобках (пример 5.1).

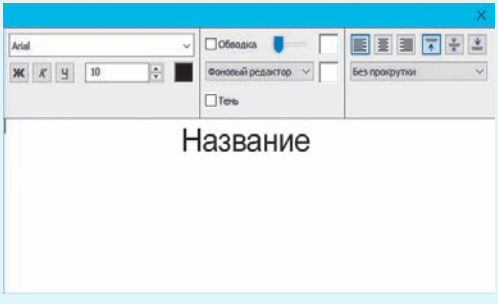
Новый текстовый клип сразу открывается в окне предпросмотра и имеет прозрачный фон, на что указывает шахматная текстура. Одновременно открывается диалоговое окно для ввода текста в клип (пример 5.2).

Текст надписи вводится с клавиатуры. Параметры шрифта изменяются инструментами, которые расположены в левой части панели окна. Размеры и положение вводимого текста контролируются в окне предпросмотра. Там же

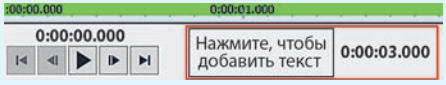
Пример 5.1. Имена нескольких текстовых клипов в разделе **Изображения**.



Пример 5.2. Диалоговое окно для ввода текста в текстовый клип напоминает окно простого текстового редактора.

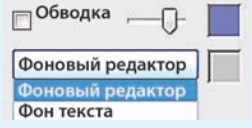


Пример 5.3. В окне предпросмотра выделены: кнопка вызова окна для ввода текста (в форме текстового поля) и поле длительности текстового клипа.



Длительность текстового клипа можно изменить после щелчка по числовому полю. Обычно вводится число секунд. Числовые значения правее числа секунд можно удалять.

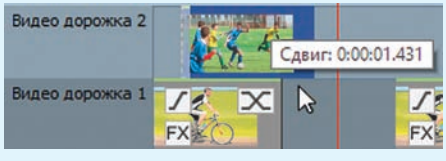
Пример 5.4. Меню кнопки **Фоновый редактор** в окне ввода текста.



Выбор пункта **Заполненный фон** в меню кнопки **Фоновый редактор** одновременно меняет и надпись на этой кнопке. Кнопка получает название **Заполненный фон**.

Пример 5.5. Полоса над **Видеодорожкой 1** для наложения текстового клипа с прозрачным фоном поверх видеоряда.

Пример 5.6. Сдвиг клипа по видеодорожке вправо-влево перетаскиванием мыши облегчается удерживанием клавиши **Shift** клавиатуры.



При перетаскивании текстового клипа около указателя мыши отображается величина сдвига.

в числовом поле отображается длительность текстового клипа (пример 5.3).

Для создания в текстовом клипе цветного фона в диалоговом окне для ввода текста используют кнопку **Фоновый редактор**. Щелчок по кнопке вызывает меню, в котором выбирается пункт **Заполненный фон**, а затем щелчком по квадратному полю правее начинается выбор цвета (пример 5.4).

Теперь уже в меню кнопки **Заполненный фон** выбор пункта **Фоновый редактор** возвращает прозрачность фону текстового клипа. В диалоговом окне для ввода текста прозрачность фона не отображается.

5.2. Вставка и наложение текстовых клипов

Текстовые клипы вставляются в видеоряд как обычные видеофрагменты.

Текстовые клипы с прозрачным фоном накладываются поверх видеоряда в режиме **Шкала времени** окна видеоряда. В этом режиме видна основная **Видеодорожка 1** и над ней — полоса для новой видеодорожки (пример 5.5). Именно в эту полосу (новую видеодорожку) текстовый клип перетаскивается мышью из раздела **Изображения** для файлов.

Положение добавленного текстового клипа можно изменить перетаскиванием изображения клипа вправо-влево по новой видеодорожке при помощи мыши (пример 5.6).

5.3. Видеопереходы между клипами

Как и между слайдами в компьютерных презентациях, между клипами можно вставлять эффекты смены, ко-

торые называются **видеопереходами.** В окне видеоряда на изображении каждого видеоклипа есть Х-образный **значок видеоперехода** (пример 5.7).

Щелчок по значку видеоперехода открывает обширное меню для выбора вида перехода к следующему клипу (пример 5.8).

Видеопереход между клипами занимает некоторое время, и в это время клипы должны показываться одновременно. Поэтому редактор предлагает выбрать способ создания перехода. Следующий клип можно сдвинуть левее на время перехода. А можно клип не сдвигать (заморозить его положение) и заполнить время перехода показом только его первого кадра.

5.4. Добавление и настройка фонограммы

Добавление и настройку музыкальной фонограммы к фильму проводят в режиме **Шкала времени** окна видеоряда. В этом режиме видео- и аудиодорожки имеют по четыре кнопки управления (пример 5.9).

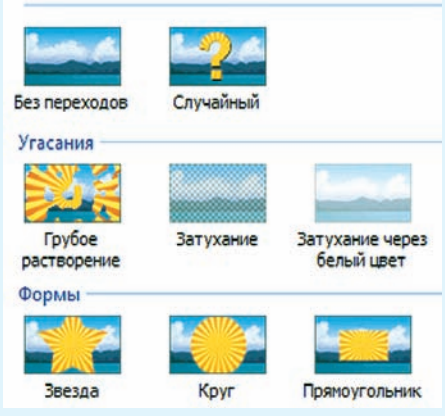
Аудиофайл (новую фонограмму) из раздела **Аудиофайлы** перетаскивают в полосу под **Аудиодорожкой 1**. Образуется новая аудиодорожка. Фонограмму по ней можно перетаскивать мышью вправо-влево.

Настройка фонограммы заключается в ее обрезке до длительности видеоряда и добавлении эффектов **Появление** и **Исчезновение**, которые в редакторе привязаны соответственно к началу и к концу фонограммы.

Пример 5.7. Значки видеоперехода на изображении видеофрагментов в видеоряде.

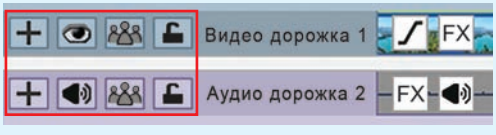
Значки видеоперехода отображаются в окне видеоряда в обоих его режимах.

Пример 5.8. Часть меню значка видеоперехода.



В меню также есть числовое поле для изменения длительности перехода.

Пример 5.9. Кнопки управления дорожками в режиме **Шкала времени** окна видеоряда. Их назначение покажут подсказки около указателя мыши.

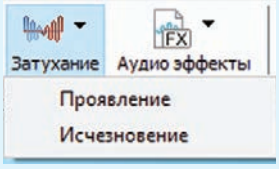


Если клипы в видеоряде имеют свои аудиодорожки, то все они расположены на **Аудиодорожке 1**. Эту аудиодорожку обычно отключают кнопкой управления **Выключить звук дорожки.**

Пример 5.10. Курсор видеоряда кнопками управления в окне предпросмотра устанавливают в конец видеоряда и в этом же окне щелкают по кнопке **Разделить**. Фонограмма в окне видеоряда делится на две части.

В окне видеоряда 2-ю часть фонограммы выделяют щелчком мыши и удаляют клавишей **Delete** клавиатуры.

Пример 5.11. Кнопка **Затухание** открывает меню с названиями эффектов.



Эффекты применяются поочередно.

Обрезка фонограммы до длительности видеоряда включает ее разделение на две части и удаление второй части (пример 5.10).

Чтобы добавить один из эффектов, фонограмму в окне видеоряда выделяют и щелкают по кнопке **Затухание** на вкладке **Видеоряд**. Название эффекта выбирается в выпадающем меню (пример 5.11). Под конец открывается диалоговое окно **Появление** или **Исчезновение**, в котором в текстовое поле вводится длительность эффекта. Практика показывает, что для длительности эффектов появления и исчезновения 2 с вполне достаточно.

1. Что такое текстовый клип?
2. Каким образом текстовые клипы вставляют в видеоряд?
3. Каким образом текстовые клипы накладывают поверх видеоряда?
4. Что такое переходы между клипами?
5. Для чего служит значок перехода на изображении клипа?
6. Как проводится добавление музыкальной фонограммы к фильму?
7. В чем состоит обрезка фонограммы до длительности видеоряда?

Упражнения

1 Загрузите в видеоредактор VideoPad файл проекта **Вода.vpj**.

2 Откройте (загрузите в видеоредактор) аудиофайл с музыкальной композицией (данная композиция лицензионных ограничений не имеет). Прослушайте ее.

3 На базе проекта **Вода.vpj** создайте и сохраните в файле **Вода.avi** видеофильм, при создании которого должны быть выполнены перечисленные требования.

1. Название фильма «Вода» должно быть размещено на фоне первого клипа.
2. Текстовый клип с надписью «Конец» на черном фоне должен быть размещен в конце фильма.
3. Между клипами должны быть установлены переходы.
4. В качестве фонограммы к фильму должна быть использована музыкальная композиция из загруженного аудиофайла.

Глава 2
ОСНОВЫ АНИМАЦИИ

§ 6. Основные понятия. Редактор для создания анимации

6.1. Анимация. Виды анимации

Человек с древности пытался запечатлеть движение в рисунке. Сегодня движение изображений можно видеть в мультфильмах, в видеофильмах, в рекламных баннерах на веб-страницах и др. А слово *анимация* известно даже маленьким детям.

Анимация (от лат. *animare* — оживить) — процесс изменения размера, положения, цвета или формы объекта с течением времени.

Анимация представляет собой последовательную демонстрацию серии изображений (кадров). Кадр отображается некоторое время, после чего исчезает, а на его месте появляется новый.

Кадры — изображения последовательных фаз движения объектов или их частей.

Анимация основана на свойстве человеческого зрения «помнить» изображение в течение некоторого времени, после того как наблюдение прекращается. Чем больше кадров содержит анимация, тем более сглаженным будет движение в процессе ее проигрывания. Для создания иллюзии непрерывного движения частота смены кадров должна быть не менее 12 кадров в секунду.

Первоначально, при подготовке кадров для анимации, каждый кадр рисовался отдельно и полностью, что отнимало много времени даже у большого

Самый ранний образец анимации создан примерно 5000 лет назад. Он представлен на кубке из необожженной глины, обнаруженном во время раскопок в Иране. На кубке изображена коза, которая подпрыгивает и срывает листья с пальмового дерева.

Каждое отдельное изображение козы на кубке представляет собой отдельный кадр.

В 1877 г. было запатентовано изобретение француза Шарля-Эмиля Рейно (1844—1918) — праксиноскоп.

Шарль-Эмиль Рейно

Праксиноскоп

Праксиноскоп представлял собой устройство из открытого цилиндра, в центре которого находилась зеркальная призма. Число граней призмы соответствовало количеству изображений-миниатюр. При быстром вращении цилиндра на видимой грани призмы создавалась иллюзия движения.

Уолт Дисней

Н. Н. Константинов

Послойную технику в мультипликации впервые применил Уолт Дисней (1901—1966) — американский художник-мультипликатор, основатель компании Walt Disney Productions.

В 1968 г. группа советских ученых во главе с Николаем Николаевичем Константиновым (советским и российским математиком, род. в 1932 г.) создала математическую модель движения животного. Вычислительная машина БЭСМ-4, выполняя программу, прорисовывала кадры мультфильма с анимацией движений кошки. Для создания кинопленки с мультфильмом каждый кадр был распечатан на принтере, роль пикселя играла буква «Ш».

В настоящее время существуют различные технологии создания компьютерной анимации. Например:

- **Запись движения.** Актеры в специальных костюмах с датчиками совершают движения, которые записываются камерами и анализируются специальным программным обеспечением. Итоговые данные о перемещении суставов и конечностей актеров применяют к трехмерным скелетам виртуальных персонажей, благодаря чему добиваются высокого уровня достоверности их движения.
- **Процедурная анимация** полностью или частично рассчитывается компьютером.
- При **программируемой анимации** движения объектов программируются с помощью браузерного языка JavaScript и языка работы с Flash-приложениями ActionScript.

коллектива художников. Затем стала использоваться послойная техника рисования объектов и фонов на прозрачных пленках, накладываемых друг на друга. Это снизило трудоемкость работ, т. к. не нужно было рисовать каждый кадр полностью. Современные анимационные технологии переведены на компьютерную основу.

Компьютерная анимация — создание анимации с помощью компьютера.

Работая над созданием компьютерной анимации, художник обычно прорисовывает начальное и конечное положение движущихся объектов, а все промежуточные состояния рассчитывает и изображает компьютер. Объекты компьютерной анимации размещаются на разных слоях (подобно прозрачным пленкам в классической анимации).

При создании компьютерной анимации могут использоваться растровые изображения (Gif-анимация) и векторные рисунки (Flash-анимация).

Выделяют два способа создания компьютерной анимации:

- покадровая анимация;
- расчетная анимация — анимация движения объектов и анимация формы.

При создании **покадровой анимации** прорисовываются все фазы движения объекта. Такая технология незаменима при создании сложной анимации с разнообразной графикой.

Анимация движения или **формы** предполагает рисование только отдельных кадров. В этих кадрах объект располагается в начале и в конце дви-

жения. Все остальные кадры — промежуточные. Изображение в них создает компьютерная программа, которая вычисляет, где и в какой момент должен находиться объект. Расчетная анимация используется для создания анимационных эффектов на веб-страницах, а также при создании рекламных, учебных и развлекательных фильмов.

6.2. Редактор Flash

Появлению компьютерной анимации способствовало развитие программ для работы с графикой.

Программы для работы с анимацией представлены в примере 6.1. Одним из наиболее популярных редакторов для создания анимации является Flash. Преимущество Flash в том, что с его помощью можно создать красивую анимацию, а файлы будут небольшого размера. Редактор Flash имеет интерфейс, многие элементы которого знакомы вам по опыту работы в графических редакторах (см. *Приложение 2*, с. 156).

В начале работы в редакторе необходимо создать новый документ (**File** → → **New...**) либо открыть уже существующий (**File** → **Open...**).

Документ, созданный в Flash, принято называть **фильмом**. В окне редактора Flash можно одновременно открывать несколько файлов с фильмами. Вкладки открытых файлов располагаются под строкой меню: 

.

Для перехода к нужному файлу достаточно щелкнуть мышью по вкладке с его именем. Звездочка справа от имени файла обозначает, что в нем сделаны изменения, которые не сохранены.

Пример 6.1. Для создания анимированных изображений существует множество программ, как платных, так и бесплатных. Например:

	Vectorian Giotto
	Adobe Animate
	Easy GIF Animator
	Pivot Stickfigure Animator

История Flash началась в 1996 г., когда компания Macromedia выпустила продукт под названием Flash.

В 2005 г. вышла версия Macromedia Flash Professional 8. В этой версии улучшена работа с графикой и анимацией. В том же году фирма Adobe купила Macromedia вместе с ее продуктами, включая Flash.

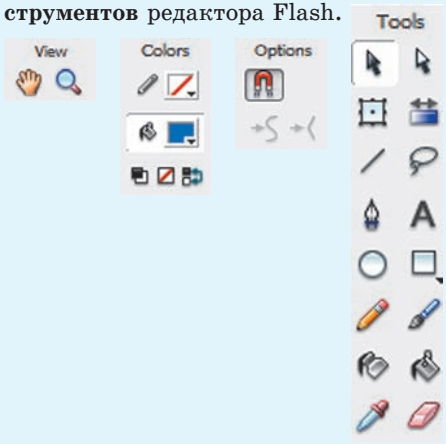
Значки различных версий редактора Flash:

	Macromedia Flash MX 2002
	Macromedia Flash Professional 8 2005
	Adobe Flash Professional CC 2015

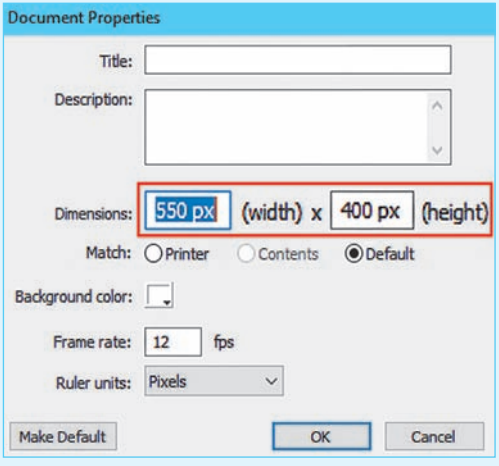
Для устройств, которые не поддерживают Flash, можно сохранить фильм в форматах HTML и GIF. В этом случае мультипликация может быть воспроизведена практически на всех устройствах.

Редактор Flash поддерживает язык сценариев (описания поведения объекта) ActionScript. Используя этот язык, можно создавать интерактивные (содержащие элементы взаимодействия с пользователем) фильмы.

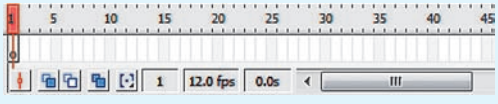
Пример 6.2. Разделы **Панели инструментов** редактора Flash.



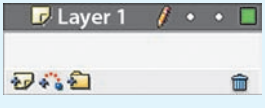
Пример 6.3. Изменение размеров монтажного стола.



Пример 6.4.



Шкала времени



Список слоев

В левой части окна редактора Flash располагается **Панель инструментов,** состоящая из четырех разделов:

- **Инструменты (Tools)** — инструменты рисования и редактирования;
- **Просмотр (View)** — способ просмотра;
- **Цвета (Colors)** — цвета обводки и заливки;
- **Параметры (Options)** — настройка свойств выбранного инструмента.

(Рассмотрите пример 6.2.)

В рабочей области редактора Flash можно выполнять операции создания и редактирования объектов (см. *Приложение 2*, с. 156). В кадр попадают только те объекты, которые расположены в пределах **монтажного стола**. Остальная часть рабочей области нужна для предварительных рисунков и для реализации эффекта постепенного входа объекта в кадр (или выхода из кадра).

Размеры монтажного стола можно изменить (пример 6.3) в окне **Свойства документа (Document Properties).**

Над рабочей областью находится **шкала времени** и **список слоев** (пример 6.4). **Шкала времени (Timeline)** предназначена для работы с кадрами. **Слои (Layers)** — компьютерный аналог прозрачных пленок, которые применяются в традиционной мультипликации.

В правой части окна размещаются дополнительные панели (пример 6.5).

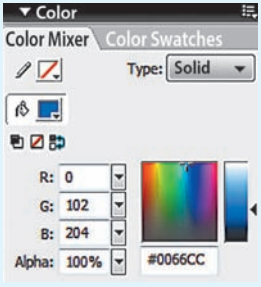
Редактор Flash поддерживает векторную графику, однако позволяет использовать и растровые изображения, импортированные из внешних файлов. При сохранении фильма в редакторе Flash обычно используют два типа файлов: **.fla** и **.swf.** Собственным форматом

Flash является формат **FLA**. В этом формате фильм сохраняется для последующего редактирования. Для реализации возможности просмотра фильма его нужно опубликовать.

Публикация — сохранение фильма в формате **SWF**. Опубликовать файл можно, выполнив команду **Файл** → → **Опубликовать (File** → **Publish)** или с помощью комбинации клавиш **Ctrl** + + **Enter**. Файл при публикации сохраняется в папку, в которой находится файл формата FLA. Опубликованный фильм можно просмотреть в любом проигрывателе Flash, а также в браузере.

Редактор Flash позволяет экспортировать результаты работы в другие форматы. Для этого необходимо выполнить команду **Файл** → **Экспорт** → **Экспорт фильма (File** → **Export** → **Export Movie)** и выбрать формат файла (пример 6.6).

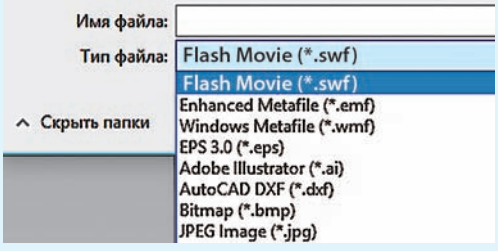
Пример 6.5. Дополнительные панели.



Смеситель цветов (Color Mixer)

Библиотека (Library)

Пример 6.6. Экспорт файлов.



1. Что называют анимацией?
2. Что представляет собой кадр?
3. С какой минимальной частотой должна происходить смена кадров, чтобы создавалась иллюзия непрерывного движения?
4. В чем заключается покадровая анимация?
5. Чем расчетная анимация отличается от покадровой?
6. В каком формате нужно сохранить фильм, чтобы его можно было редактировать?
7. Что значит опубликовать фильм?

Упражнения

1 Откройте редактор Flash. Создайте новый документ. Используя материал § 6 и *Приложения 2* (с. 156), изучите интерфейс окна редактора. Выполните:

1. В меню выберите **Вид** → **Сетка** → **Показать сетку (View** → **Grid** → → **Show Grid)**. Как изменилась страница в Рабочей области?
2. Установите размеры монтажного стола 800 px (width) x 200 px (height).
3. Изучите инструменты раздела **Просмотр (View)** на **Панели инструментов**. С помощью инструмента **Рука (Hand Tool)** ( ) переместите страницу в Рабочей области. Увеличьте видимый размер страницы и

уменьшите его с помощью инструмента **Лупа (Zoom Tool)**. Настройте увеличение (уменьшение) масштаба в разделе **Параметры** ( ).

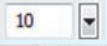
4. Откройте (закройте) дополнительные окна **Смеситель Цветов** и **Библиотека**.

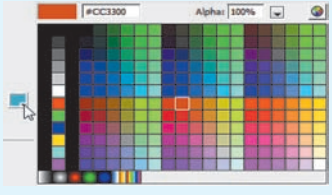
2 Откройте в редакторе Flash файл. Сохраните файл под новым именем. Выполните публикацию фильма. Просмотрите фильм в проигрывателе Flash.

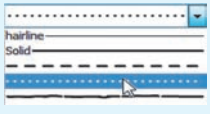
§ 7. Создание изображений и редактирование объектов

Пример 7.1. Рисование линий.

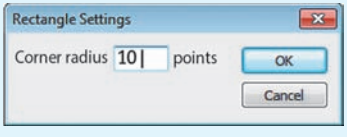
Перед рисованием линии на **Панели свойств** выбрать:

1. Толщину: 

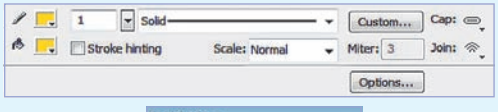
2. Цвет: 

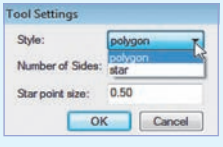
3. Стиль: 

Пример 7.2. Рисование прямоугольника с закругленными углами.



Пример 7.3. Диалоговое окно **Tool Settings**.





7.1. Создание изображений

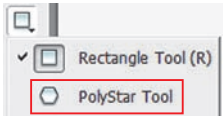
Основным анимируемым объектом редактора Flash является векторное изображение. Создание векторных изображений в Flash имеет много общего с аналогичным процессом в векторных редакторах. Рассмотрим особенности использования инструментов рисования редактора Flash.

Для рисования линий в редакторе Flash используется инструмент **Линия (Line)**. Определить цвет, стиль и толщину можно на **Панели свойств** (пример 7.1). Линии с углом наклона, кратным 45°, рисуются при нажатой клавише Shift.

Овал можно нарисовать с помощью инструмента **Овал (Oval)**. Этот инструмент также используется для рисования круга и окружности (при нажатой клавише Shift).

Для рисования прямоугольников используют инструмент **Прямоугольник (Rectangle)**. Чтобы задать радиус закругления углов прямоугольника, нужно на **Панели инструментов** в разделе **Параметры** выбрать и в диалоговом окне **Настройки прямоугольника (Rectangle Settings)** ввести значение радиуса в пикселях (пример 7.2).

Для рисования многоугольников и звезд нужно выбрать инструмент **Многоугольник/Звезда (PolyStar)**, развернув список инструмента :



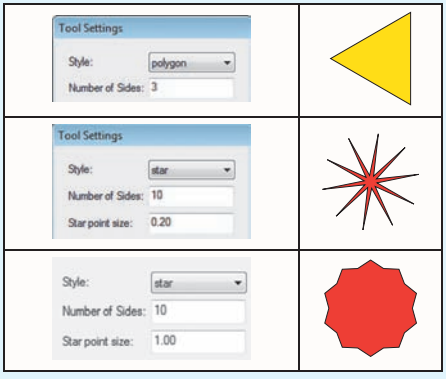
Когда инструмент активен, на **Панели свойств** находится кнопка Options..., которая вызывает окно настройки инструмента (пример 7.3). В нем можно выбрать тип фигуры (многоугольник/звезда), задать количество сторон (лучей) и их размер (пример 7.4).

Для рисования линий и кривых используется инструмент **Перо (Pen)** (пример 7.5). С помощью этого инструмента легко рисовать ломаную линию. Щелчками мыши рисуется отрезок или контур, состоящий из отрезков прямых линий, соединенных угловыми (опорными) точками.

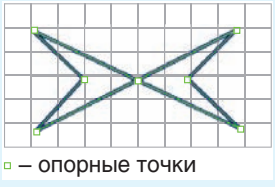
Для рисования контуров произвольной формы используется инструмент **Карандаш (Pencil)**. Для инструмента **Карандаш** можно выбрать различные режимы рисования (пример 7.6):

- **Выпрямление (Straighten)** позволяет преобразовать исходное изображение в одну из геометрических фигур (пример 7.7);
- **Сглаживание (Smooth)** сглаживает линии. Степень сглаживания линий задается на **Панели свойств** в поле Smoothing 85;
- **Рисунок чернилами (Ink)** похож на режим **Smooth**. Степень сглаживания незначительна и не изменяется.

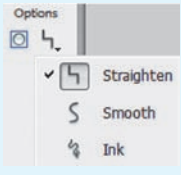
Пример 7.4. Рисование многоугольников и звезд.



Пример 7.5. Рисование ломаной линии инструментом **Перо**.



Пример 7.6. Выбор режима рисования для инструмента **Карандаш**.



Пример 7.7. Использование инструмента **Карандаш** в режиме **Выпрямление**.

Во время рисования	После завершения рисования
	
	

Пример 7.8. Инструменты выделения:

— **Выбор (Selection);**

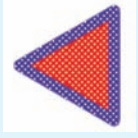
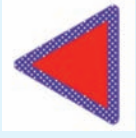
— **Выбор подобласти (Subselection);**

— **Петля (Lasso);**

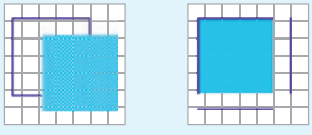
— **Произвольная трансформация (Free Transform);**

— **Перо (Pen Tool).**

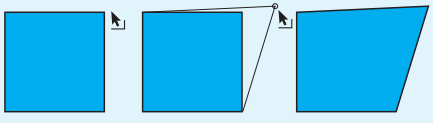
Пример 7.9. Выделение объектов.

Весь объект	Заливка
	
Часть обводки	Все части обводки (с Shift)
	

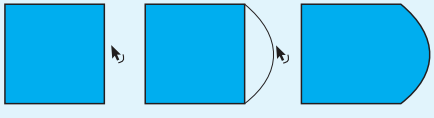
Пример 7.10. Использование инструмента .



Перемещение



Изменение положения вершин



Изменение контура

7.2. Редактирование изображений

При подготовке кадров анимации приходится редактировать существующие объекты. Свойства обводки и заливки (цвет, форма, взаимное расположение) могут изменяться пользователем независимо друг от друга. Заливку и обводку можно удалять по отдельности.

Прежде чем выполнить какие-либо действия с объектом, его необходимо выделить. Редактор Flash обладает большим спектром инструментов выделения (пример 7.8). Основной инструмент выделения — **Выбор (Selection).** С его помощью можно выделить весь объект, заключив его в прямоугольник. Тот же результат получится, если выполнить двойной щелчок по объекту. Несколько объектов выделяются при нажатой клавише Shift. Выделенный объект покрывается мелкой сеткой. Щелчком можно выделять заливку или обводку по отдельности (пример 7.9).

С помощью инструмента можно выполнять следующие операции редактирования объектов:

- **Перемещение.** Выделить объект. При появлении крестика из двунаправленных стрелок переместить объект.
- **Изменение положения вершин.** Подвести курсор к вершине и переместить вершину.
- **Изменение контура.** Не выделяя контур, подвести курсор к контуру и переместить в нужном направлении.

(Рассмотрите пример 7.10.)

Многие операции изменения объектов могут быть выполнены с помощью инструмента **Произвольная транс-**

формация (Free Transform). При работе с инструментом после выделения объекта на **Панели инструментов** становятся доступными кнопки выбора режимов:

- **Поворот и наклон (Rotate and Skew);**
- **Масштабирование (Scale);**
- **Искажение (Distort);**
- **Изгиб (Envelope).**

(Рассмотрите пример 7.11.)

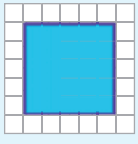
Пример 7.12 иллюстрирует изменение объекта при использовании инструмента **Произвольная трансформация** в разных режимах. Трансформация выполняется с помощью маркеров, расположенных на выделяющей рамке. Каждый маркер связан с определенной операцией. Разным маркерам соответствует свой вариант указателя мыши.

С помощью команд меню **Изменить (Modify)** можно выполнить операции над объектами:

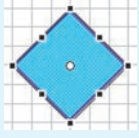
- Преобразование (изменение размера, поворот, отражение и др.) — **Трансформация (Transform);**
- Группировку (объединение нескольких объектов в один) — **Группировка (Group);**
- Выравнивание (по горизонтали и по вертикали, относительно границ объектов и относительно центра) — **Выравнивание (Align).**

Пример 7.11. Кнопки выбора режима для инструмента **Произвольная трансформация.**

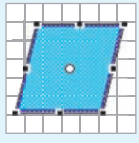
Пример 7.12. Использование инструмента **Произвольная трансформация.**



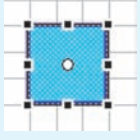
Исходное изображение



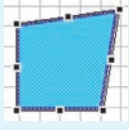
Поворот



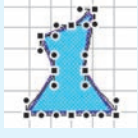
Наклон



Масштабирование



Искажение



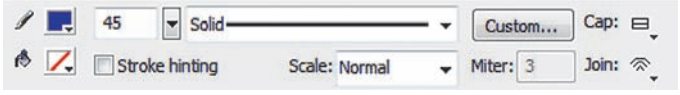
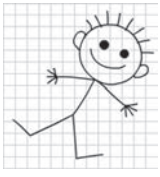
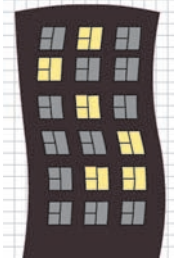
Изгиб

1. Какие инструменты редактора Flash используются для рисования?
2. В каком режиме инструмента **Карандаш** рисуются геометрические фигуры?
3. Для чего предназначен инструмент **Выбор**?
4. Какие преобразования объекта можно выполнить с помощью инструмента **Произвольная трансформация**?

Упражнения

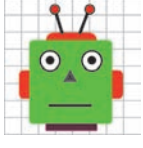
1 Откройте файл. Дополните изображение автомобиля. При рисовании окна используйте градиентную заливку. Скопируйте изображение окна, примените к копии операцию отражения. Сохраните изменения в файле.

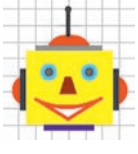
2 Создайте изображения, воспользовавшись рекомендациями.

Изображения	Рекомендации
	Используйте инструменты и с настройками:  Примените инструмент для преобразования линий в дуги либо нарисуйте дуги как часть окружности.
	Для рисования используйте инструменты и . Инструмент применяйте в различных режимах. Корректируйте изображение с помощью инструмента .
	Копирование окон выполняйте, удерживая клавишу Shift. Для изгиба дома используйте инструмент . Скорректируйте контур с помощью инструмента .

Сохраните изображения в формате FLA.

3 Нарисуйте одного из роботов. Для рисования используйте инструменты , , и инструмент редактирования . Сохраните изображение с именем, соответствующим эмоции робота.





§ 8. Слои. Библиотека объектов. Импорт объектов

8.1. Работа со слоями

Слои — важнейший элемент анимации. Применение слоев позволяет создавать сложные многоплановые сцены фильма, редактируя каждый объект на отдельном слое. Один из слоев может использоваться в качестве фона, другой содержать анимированные объекты, а третий — элементы звукового сопровождения фильма (пример 8.1).

Имена слоев показаны слева от временной шкалы. После создания файла в списке слоев находится один слой с именем **Layer 1 (Слой 1)**.

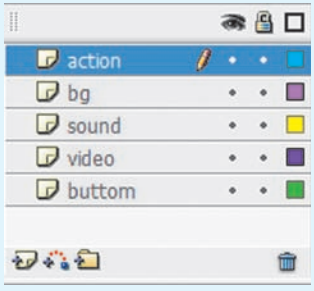
Для создания нового слоя требуется выделить тот слой, над которым вы хотите поместить новый, а затем нажать кнопку **Вставить слой (Insert Layer)**. Новый слой появляется в списке слоев над выделенным слоем, как видно из примера 8.2.

Новому слою присваивается имя **Layer** с указанием порядкового номера. Это имя обычно заменяют именем, поясняющим назначение или содержание слоя.

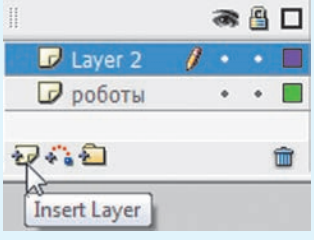
Для удобства работы со слоями в Flash реализована возможность хранения каждого набора взаимосвязанных слоев в отдельной папке слоев (пример 8.3).

Все слои абсолютно прозрачны. Объекты, расположенные на различных слоях, визуально воспринимаются как элементы единой сцены. Объект, находящийся на верхнем слое, заслоняет

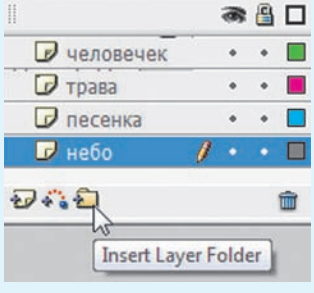
Пример 8.1. Применение слоев.

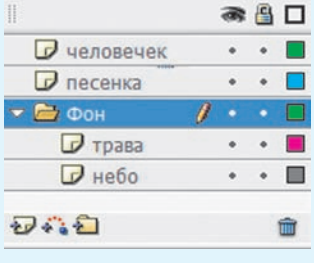


Пример 8.2. Создание нового слоя.

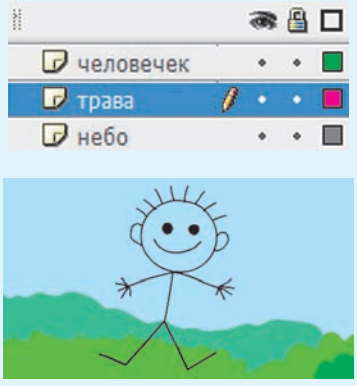


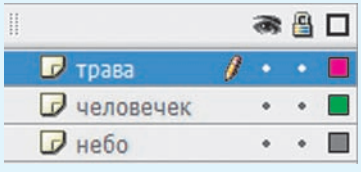
Пример 8.3. Создание папки слоев.



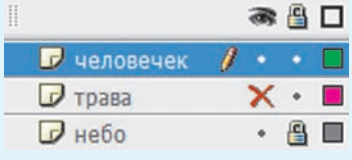


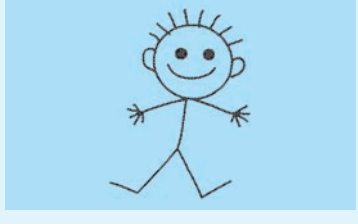
Пример 8.4. Отображение слоев в зависимости от их расположения.





Пример 8.5. Отображение слоев с различными свойствами.





объекты, находящиеся в той же позиции на нижних слоях (пример 8.4).

Можно изменять порядок расположения слоев. Объекты одного слоя редактируются независимо от объектов других слоев. При необходимости можно одновременно выбрать объекты из разных слоев и работать с ними как с единым целым. Например, изменить их цвет.

Удалить слой (папку) можно щелчком по кнопке **Удалить слой (Delete Layer)**, расположенной в нижнем правом углу **Панели управления слоями**. При удалении папки удаляются также и входящие в нее слои.

Значки справа от имени слоя отображают его свойства:

- **Активность.** В этом слое объекты создают или редактируют. Слой выделяется в списке цветом и помечается значком .
- **Видимость.** Объекты скрытого слоя не видны, слой помечается .
- **Блокировка.** Слой помечается в списке значком .

(Рассмотрите пример 8.5.)

На заблокированном или скрытом слое нельзя радактировать и создавать объекты. При создании анимации для нередактируемых слоев устанавливается блокировка и отключается видимость.

8.2. Библиотека объектов

При создании анимации возникает необходимость использовать некоторые объекты несколько раз. Для хранения таких объектов в редакторе Flash предназначена **Библиотека (Library).**

Объекты, помещенные в библиотеку, называют **символами**. Использование символов существенно ускоряет процесс разработки фильма.

Один раз созданное изображение можно многократно использовать как в одном, так и в разных фильмах.

Существуют три типа символов:

1. **Графика (Graphic).** Содержит изображение одного кадра.

2. **Кнопка (Button).** Содержит кнопки, реагирующие на действия пользователя и управляющие воспроизведением фильма.

3. **Клип (MovieClip).** Может содержать анимацию с любым количеством кадров.

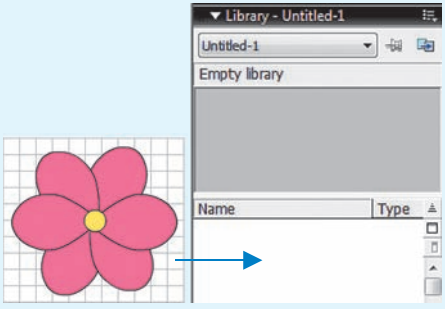
Для преобразования объекта в символ нужно нажать клавишу **F8** или перетащить изображение в окно **Библиотека**. Затем в области просмотра верхней части панели **Библиотека** можно увидеть изображение символа, а в списке символов — имя символа (пример 8.6).

Копию символа, помещенную в рабочую область, называют **экземпляром**. Экземпляры могут значительно отличаться от самого символа. При редактировании экземпляра символ не изменяется. И напротив, любые изменения символа приводят к соответствующим изменениям всех его экземпляров.

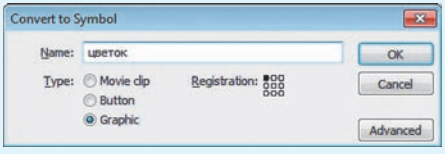
Каждый экземпляр имеет собственные свойства, которые могут редактироваться без изменения соответствующих свойств символа. Так, можно изменять цвет и прозрачность экземпляра, переопределять его тип (например, преобразовать графический символ

Пример 8.6. Создание символа типа **Графика**.

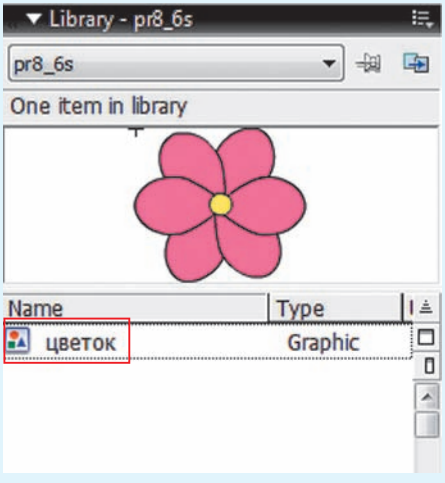
1. Перетащить выделенное изображение в список символов панели **Библиотека**:



2. В окне **Convert to Symbol** выбрать тип **Graphic** и дать имя символу:

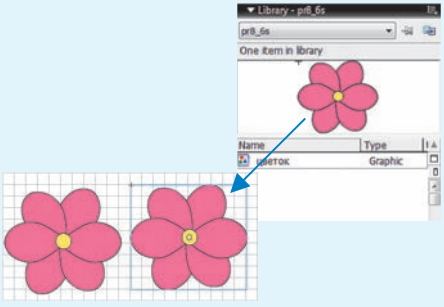


3. Просмотреть содержимое библиотеки:

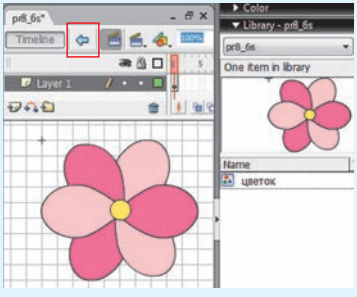


Пример 8.7. Создание экземпляров символа.

Перетащить изображение символа с панели **Библиотека** на холст.

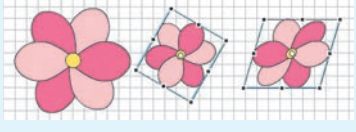


Пример 8.8. Редактирование символа.

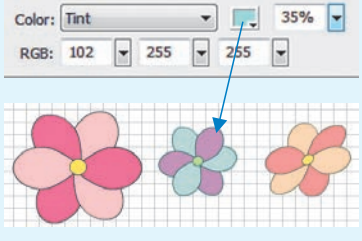


Пример 8.9. Редактирование экземпляра.

1. С помощью инструментов трансформации.



2. Изменение цвета на панели **Properties**.



в кнопку). Можно также наклонять, вращать или масштабировать экземпляр без того, чтобы воздействовать на символ.

При выделении экземпляра вокруг него появляется голубая рамка (пример 8.7).

Редактирование символа осуществляется в специальном режиме, перейти в который можно с помощью двойного щелчка по изображению символа на панели **Библиотека**. Для выхода из этого режима необходимо нажать кнопку ⇦, расположенную над списком слоев (пример 8.8).

Для редактирования свойств экземпляра используются инструменты трансформации и инструменты **Панели свойств (Properties)** (пример 8.9).

Преобразовать экземпляр символа в обычную векторную графику можно, выбрав команду **Break Apart** из контекстного меню экземпляра или нажав комбинацию клавиш Ctrl + B.

8.3. Импорт и использование объектов

Редактор Flash поддерживает многие форматы для импорта изображений — JPEG, GIF, PNG и PSD.

Существуют следующие варианты импорта изображений (меню **Файл** → → **Импорт**):

- непосредственно в рабочую область **(Import to Stage)**;
- в библиотеку **(Import to Library)**.

Импортированные в библиотеку изображения представляют собой символы.

Для импортирования изображений с прозрачностью фона рекомендуется использовать формат PNG (пример 8.10).

Импортированные растровые изображения преобразуются в векторные для последующего редактирования с помощью команды **Изменить (Modify)** → → **Bitmap** → **Trace Bitmap** (пример 8.11).

Размер импортированных изображений может не соответствовать размеру монтажного стола. Для устранения этого несоответствия изменяют либо размер монтажного стола, либо размер изображения.

Выбор команды **Открыть внешнюю библиотеку (Open External Library)** меню **Файл (File)** → **Import (Импорт)** позволяет открывать библиотеку любого файла Flash (формата .fla) и использовать ее символы.

Пример 8.10. Импорт изображений разных форматов.

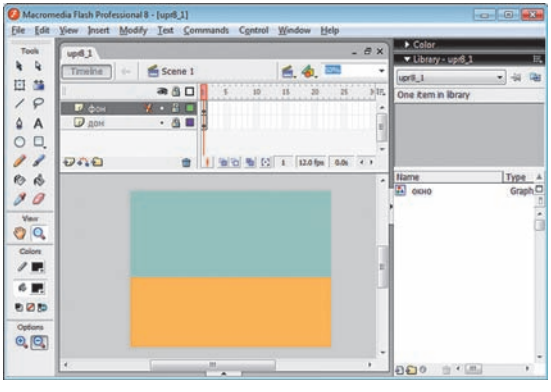
Пример 8.11. Редактирование векторного изображения, преобразованного из растрового.

1. С какой целью используют слои?
2. Для чего предназначена библиотека объектов?
3. Что представляет собой символ?
4. Как преобразовать объект в символ?
5. Что такое экземпляр символа?
6. Как перейти в режим редактирования символа?
7. Куда можно импортировать изображения в редакторе Flash?

Упражнения

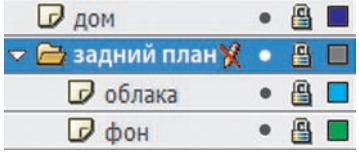
1 Откройте файл и выполните задания.

1. Измените порядок следования слоев (слой «дом» перетащите вверх). Уберите видимость слоя «дом».
2. Между слоями «дом» и «фон» создайте слой «облака». Нарисуйте в этом слое облако и преобразуйте в символ. Разместите несколько экземпляров символа «облако» на



слое. Преобразуйте экземпляры (размер, поворот, наклон, цвет). Откройте видимость слоя «дом».

3. Создайте в списке слоев папку «задний план» и перетащите в нее слои «фон» и «облака». Установите блокировку папки.



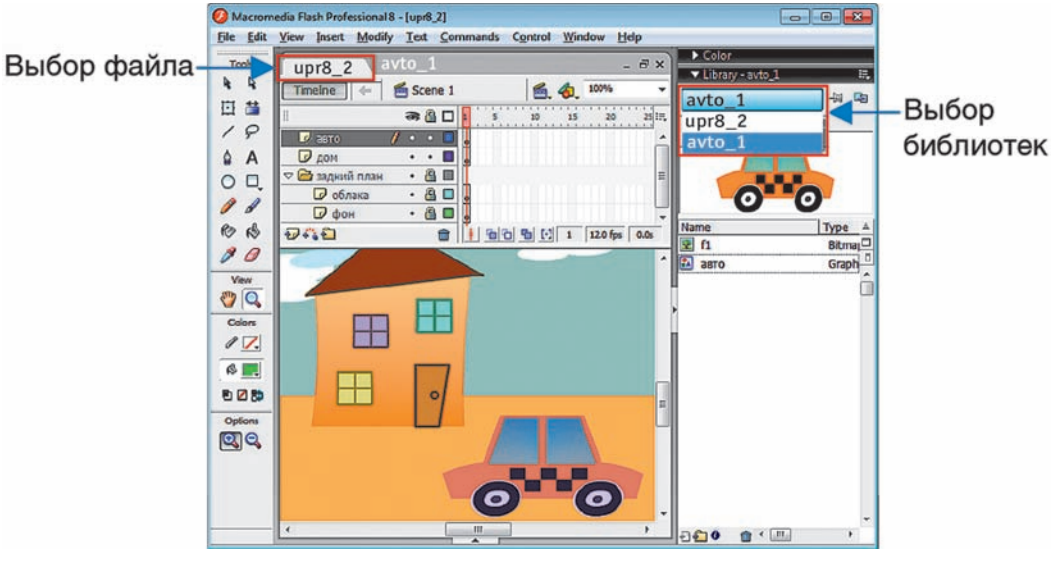
4. Активизируйте слой «дом», снимите его блокировку. Поместите на слой из библиотеки несколько экземпляров символа «окно». Измените цвет окон, используя возможности панели **Properties**.

5. Сохраните результат работы в файл с именем `upr8_2.fla`.

2 В редакторе Flash откройте одновременно файл `upr8_2.fla` и файл с изображением автомобиля, полученный в результате выполнения упражнения 1 после § 7. Выполните перечисленные действия:

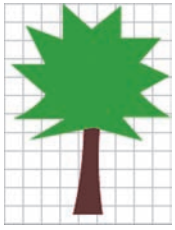
1. Создайте символ изображения автомобиля на новом слое.

2. Возвратитесь в файл `upr8_2.fla` и, выбрав библиотеку файла с автомобилем, создайте экземпляр автомобиля. Измените цвет экземпляра.



3. Сохраните изменения в файле.

3* В файле upr8_2.fla добавьте новые слои с изображениями и преобразуйте их в символы:



Дополните изображение в upr8_2.fla экземплярами символов. Измените символы в соответствии с рисунком. Сохраните изменения в файле.

§ 9. Покадровая анимация

Основным инструментом при создании анимации является шкала времени. С ее помощью можно выполнять различные операции с кадрами.

На шкале времени каждому слою соответствует строка с сеткой. Каждой ячейке соответствует отдельный кадр. Числа над шкалой обозначают номера кадров. На кадр, находящийся на монтажном столе, указывает **маркер кадра** — красный прямоугольник с линией (пример 9.1).

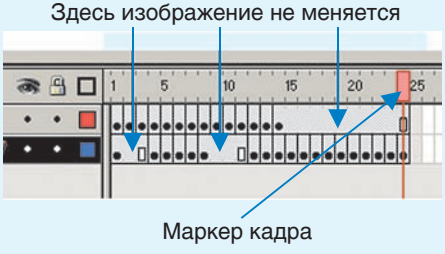
Кадры, содержимое которых определяется автором анимации, называются **ключевыми**.

Отображение кадров на шкале времени зависит от их предназначения.

(Рассмотрите пример 9.2.)

При выполнении операций с кадрами можно использовать команды контекстного меню кадра, а также «горячие» клавиши:

Пример 9.1. Отображение кадров на шкале времени.



Пример 9.2. Отображение кадров на шкале времени.

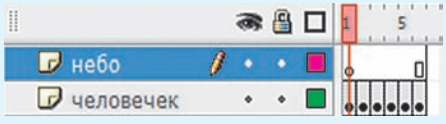
	Ключевой кадр с содержимым (редактируется, является исходным)
	Ключевой кадр без содержимого
	Простые кадры (продлевают видимость предшествующего ключевого кадра)
	Промежуточные кадры (отображают трансформацию объекта между двумя ключевыми кадрами)

Пример 9.3. Вставка ключевого кадра.

1. Выделить ячейку сетки.

2. Нажать F6.

Пример 9.4. Создание фонового слоя.



При создании покадровой анимации рекомендуется придерживаться следующих правил:

1. Выполняя сжатие и растяжение, сохраняйте объем (удлиняя объект, не забывайте сужать его).

2. Объект не должен слишком резко останавливаться и замирать — количество кадров должно быть таким, чтобы движение объекта было плавным.

3. Медленный выход и медленный вход.

4. Верный расчет времени — важно задать достаточно времени, чтобы подготовить зрителя к ожиданию действия, самому действию и реакции на действие.

Обычно на монтажном столе находится один кадр. Чтобы упростить размещение и редактирование покадровой анимации, можно просматривать на монтажном столе два и несколько кадров одновременно. Такой режим называется режимом калькирования.

- Insert Keyframe, F6 — вставка копии ключевого кадра;
- Insert Blank Keyframe, F7 — вставка пустого ключевого кадра;
- Insert Frame, F5 — вставка простого кадра;
- Clear Keyframe, Shift + F6 — очистка ключевого кадра;
- Remove Frames, Shift + F5 — удаление кадра.

Перед вставкой кадра необходимо предварительно выделить ячейку сетки, предназначенную для размещения нового кадра (пример 9.3). Операции выделения группы кадров выполняются так же, как и операции выделения группы других объектов.

Для изменения положения кадра в шкале времени достаточно перетащить его в требуемую позицию.

Анимация, полностью состоящая из ключевых кадров, называется **покадровой анимацией.**

При создании покадровой анимации для каждого ключевого кадра устанавливается длительность проигрывания.

Чем больше кадров в покадровой анимации, тем естественнее движения персонажей. Чтобы движения объектов в анимации не были резкими, а фильм слишком коротким или быстрым, можно добавить:

- ключевые кадры с промежуточным положением анимируемого объекта;
- простые кадры после каждого ключевого кадра.

Для добавления к анимации фона нужно создать еще один слой. При создании нового слоя сразу создается пустой ключевой кадр и простые кадры

по длине уже готовой анимации. Одного ключевого кадра достаточно, т. к. фон статичен (пример 9.4).

Перед публикацией анимацию нужно протестировать — выбрать в меню редактора команду **Контроль → Тестирование фильма (Control → Test Movie)**.

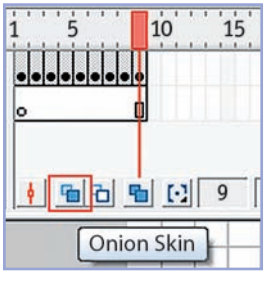
В примере 9.5 показан процесс создания покадровой анимации.

Преимуществом покадровой анимации является ее естественность, поскольку каждый следующий кадр не похож на предыдущий.

Недостатки покадровой анимации:

- при необходимости изменения всей анимации нужно изменять каждый кадр;
- занимает большой объем, т. к. приходится хранить информацию о каждом кадре.

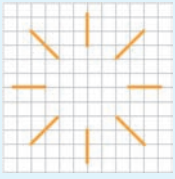
В Flash есть инструменты для калькирования анимации. Например, функция **Луковая шелуха (Onion Skin)**, позволяющая аниматору просматривать любое количество последовательных кадров.



После выбора режима калькирования содержимое выделенного в данный момент кадра отображается в полном цвете, а содержимое соседних кадров отображается в виде полупрозрачного шлейфа.

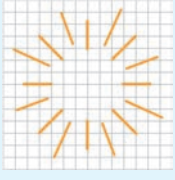
Пример 9.5. Создание покадровой анимации.

1. Открыть файл sun.fla с изображением лучей солнца.

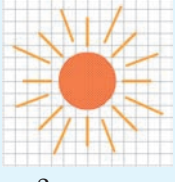


2. Создать копию кадра.

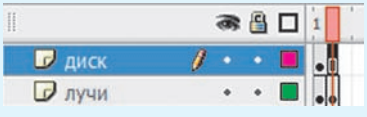
3. Изменить второй кадр, дополнив его копией лучей. Для изменения копии применить инструмент (поворот, масштабирование). Совместить центры копии и исходного изображения (**Modify → Align → Horizontal Center + Vertical Center**):



4. Создать новый слой «диск» с изображением диска солнца:



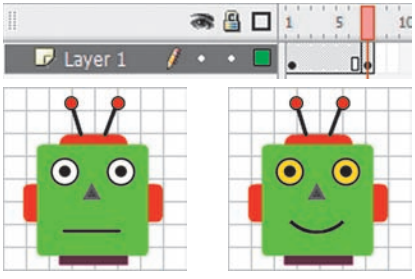
5. В кадре 2 слоя «диск» создать простой кадр:



6. Сохранить изменения в файле.
7. Протестировать анимацию.
8. Опубликовать фильм.

1. Для чего предназначена шкала времени?
2. Какие кадры называются ключевыми?
3. Как создается дубль ключевого кадра?
4. Какую анимацию называют покадровой?
5. Какую комбинацию клавиш используют для тестирования анимации?

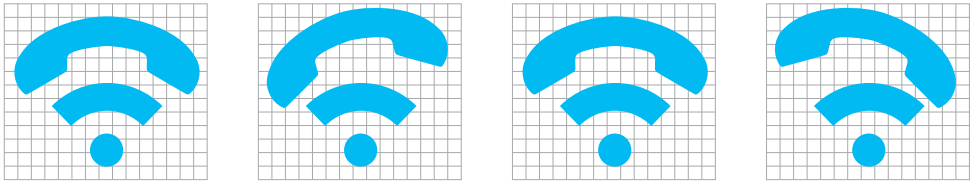
Упражнения



1 Откройте файл. С помощью покадровой анимации измените эмоцию робота, как показано на рисунке.

Сохраните файл. Опубликуйте получившуюся у вас анимацию.

2 Откройте файл (результат выполнения задания 1 из упражнения 2 после § 7). Создайте покадровую анимацию из четырех ключевых кадров.



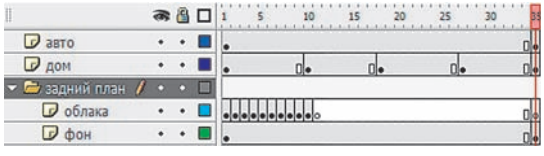
Сохраните изменения в файле. Опубликуйте анимацию.

3 Откройте файл (результат выполнения упражнения 1 после § 8). Создайте покадровую анимацию из пяти ключевых кадров в слое «дом».

1. В кадрах 2 и 4 измените цвет окон.
2. Добавьте простые кадры в соответствующих слоях.

4 Дополните анимацию, полученную после выполнения упражнения 3, анимацией перемещения облаков.

Сохраните анимацию в файле `upr9_4.fla`. Опубликуйте анимацию.



5* Добавьте к анимации, полученной после выполнения упражнения 4, анимацию солнца из примера 9.5.

§ 10. Анимация движения

10.1. Прямолинейное движение

При работе над покадровой анимацией кадры, содержащие промежуточные фазы движения объектов, вы создавали сами. Создание промежуточных кадров можно доверить компьютеру. В этом случае достаточно задать состояние объекта анимации в начале и в конце движения, а все промежуточные фазы движения рассчитает программа Flash.

Такая анимация называется **анимацией движения** и применима только к символам. На слое может находиться один анимированный символ.

Изображения в анимации движения находятся в ключевых кадрах. Промежуточные кадры хранят ссылки на первый ключевой кадр, что позволяет сократить размер файла с анимацией.

Включить/выключить режим анимации движения можно с помощью команд контекстного меню начального кадра:

- **Создать анимацию движения (Create Motion Tween);**
- **Удалить движение (Remove Tween).**

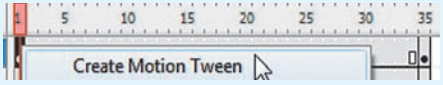
(Рассмотрите пример 10.1.)

Самая простая анимация движения — движение по прямой (пример 10.2).

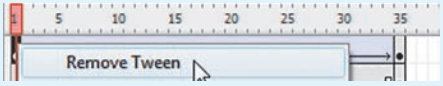
После выполнения команды **Создать анимацию движения** между ключевыми кадрами появляется сплошная стрелка, расположенная на лилово-голубом фоне. Промежуточные кадры отражают последовательность фаз

Пример 10.1. Использование контекстного меню начального кадра.

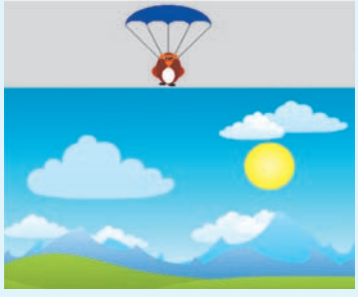
1. Создание движения:



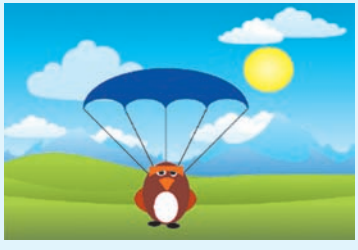
2. Удаление движения:



Пример 10.2. Создание анимации прямолинейного движения.

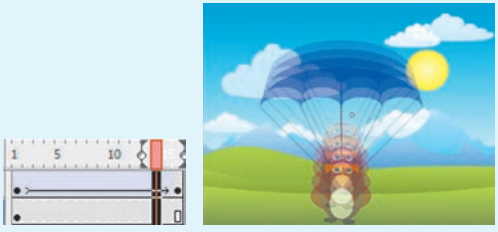


1. Поместить фоновый рисунок и парашютиста на разные слои.
2. Изображение парашютиста преобразовать в символ (F8).
3. Выделить в слое с фоном кадр 45 и добавить простые кадры (F5).
4. Выделить в слое с парашютистом кадр 45 и преобразовать его в ключевой (F6).
5. Изменить размеры и положение парашютиста в кадре 45:



6. Создать анимацию движения парашютиста по прямой.

Пример 10.3.



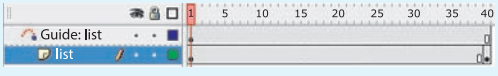
Появление вместо стрелки штриховой линии означает, что допущена ошибка и анимация не создана.

Пример 10.4. Создание анимации движения по траектории.

1. Нарисовать или импортировать изображение.

2. Создать направляющий слой и изобразить на нем траекторию.

3. В кадр 40 слоя с изображением вставить копию ключевого кадра (**F6**), а в слой с траекторией добавить простые кадры (**F5**).

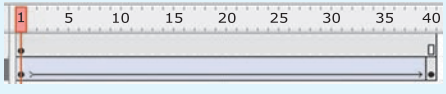


4. Убедиться, что на Панели инструментов активен параметр **Защелка:** 

.

5. Переместить изображение в ключевых кадрах в положение, когда точки трансформации совпадают с концами линии траектории.

6. В слое с изображением включить режим анимации движения.



движения (пример 10.3). При создании анимации движения важно, чтобы:

1. Начальная и конечная фазы анимации были получены из точных копий одного и того же объекта.

2. В начальном и в конечном ключевых кадрах находились только единственные объекты.

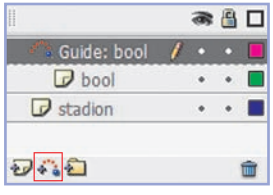
10.2. Движение по траектории

Для создания анимации движения по заданной траектории необходим специальный **направляющий слой.**

Направляющий слой размещается над слоем с анимируемым объектом. Для добавления направляющего слоя нужно:

1. Выделить слой с анимируемым объектом.

2. Нажать кнопку **Добавить движение (Add Motion Guide):**



На направляющем слое изображается траектория, по которой будет перемещаться анимируемый объект. При просмотре фильма линия траектории не отображается. В слое с анимируемым объектом задается анимация движения, как в случае прямолинейного движения (пример 10.4).

Для привязки объекта к траектории необходимо:

1. Включить параметр **Защелка (Snap)** на **Панели инструментов.**

2. Совместить точку трансформации объекта (кружок в центре) с началом линии траектории (в первом кадре) и концом (в конечном кадре).

Если объект не привязать к траектории, то он будет двигаться прямолинейно.

При создании анимации движения нескольких объектов для каждого слоя с объектами создается свой слой направляющих. Также можно привязать несколько слоев к одному слою направляющих, чтобы несколько объектов двигались по одной и той же траектории.

1. К каким объектам можно применить анимацию движения?
2. Как создать анимацию движения?
3. Как удалить анимацию движения?
4. Для чего предназначен направляющий слой?
5. Как создается направляющий слой?
6. Как связать траекторию с объектом движения?

Упражнения

1 Откройте файл. Выполните пример 10.2. Сохраните изменения в файле `upr10_1.fla`. Опубликуйте анимацию.

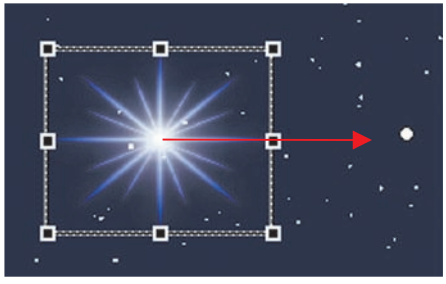
2 Внесите изменения в файл `upr9_4.fla` (результат выполнения упражнения 4 после § 9).

1. В слое «облака» замените покадровую анимацию на анимацию движения.
2. Создайте анимацию прямолинейного движения автомобиля слева направо.
3. Добавьте второй экземпляр автомобиля и создайте прямолинейную анимацию движения автомобиля справа налево.

Сохраните созданную анимацию в файле `upr10_2.fla`. Опубликуйте анимацию.

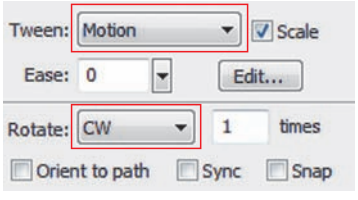
3 Откройте файл. Создайте анимацию движения звездочки по кругу, выполняя данные ниже рекомендации.

1. С помощью инструмента перенесите точку трансформации (центр будущего вращения) на некоторое расстояние.



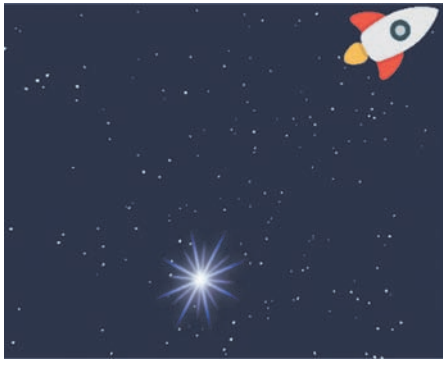
2. В кадр 30 вставьте копию ключевого кадра.

3. Перейдите на первый кадр и откройте **Панель свойств.** В списке **Определение (Tween)** выберите **Движение (Motion).** В списке **Поворот (Rotate)** выберите принудительное вращение **по (CW)** или **против (CCW) часовой стрелки.**



4. Запустите анимацию на просмотр. Сохраните изменения в файле. Опубликуйте анимацию.

4 Дополните анимацию из упражнения 3 анимацией движения ракеты. Сохраните изменения в файле `upr10_4.fla`. Опубликуйте анимацию.



5 Создайте анимацию движения по траектории. Выполните импорт изображений из файлов.

Фоновое изображение	Анимируемое изображение	Траектория движения
		
		

§ 11. Анимация формы

Анимация формы — плавное изменение объекта анимации.

При создании анимации формы объектом является не экземпляр, как при анимации движения, а обычное векторное изображение. Количество примитивов в изображении может быть различным в начале и в конце анимации.

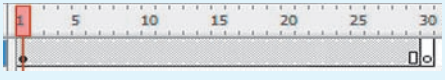
В процессе анимации формы изображение может разделиться на несколько независимых фрагментов, каждый из которых будет постепенно трансформироваться. Или, наоборот, несколько независимых изображений при анимации, постепенно меняя облик (размеры, цвет, форму), могут стать частями единого изображения. Поэтому анимация формы лучше всего подходит для простых изображений без обводки.

В рамках анимации формы можно также изменять положение и цвет объектов (пример 11.1). Перемещение

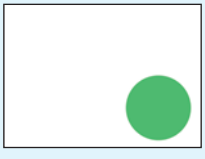
Пример 11.1. Создание анимации формы.

1. В первом кадре нарисовать квадрат:

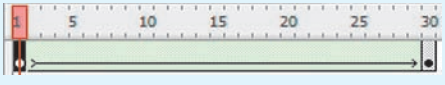
2. Добавить пустой ключевой кадр в кадре 30 (F7)



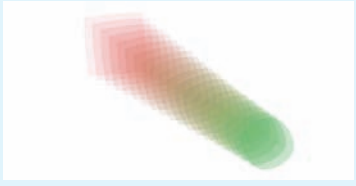
и нарисовать в нем круг:



3. Создать анимацию формы:



Пример 11.2. Отображение промежуточных фаз.



Переход одной формы в другую не всегда предсказуем. Если исходный и конечный объекты содержат несколько фигур, то трудно предположить, как будет происходить трансформация.

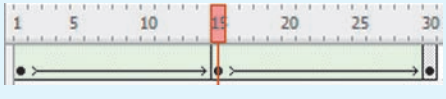
Пример 11.3. Использование растровых изображений.

1. Импортировать изображение на монтажный стол.

2. Преобразовать символ в обычную графику.

3. В кадрах 15 и 30 добавить копии ключевых кадров. Изменить в кадре 15 изображение:

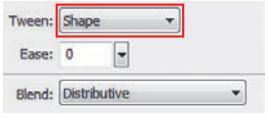
4. Создать анимацию формы.



объектов при анимации формы всегда прямолинейно. Для создания анимации формы требуется:

1. Выбрать начальный ключевой кадр (или любой кадр между двумя ключевыми).

2. На **Панели свойств** выбрать **Форма (Shape)**:



Между ключевыми кадрами после создания анимации формы должна появиться сплошная стрелка на салатовом фоне. Промежуточные кадры будут отражать последовательность фаз изменения формы (пример 11.2). При задании отрицательных значений в поле параметра **Замедление (Ease)** анимация будет идти ускоренно, а при вводе положительных значений этого параметра — замедленно.

Вам уже известно, что импортированные растровые изображения представляют собой символы. Их можно использовать при создании анимации формы, предварительно преобразовав в обычную графику (пример 11.3).

При анимации объектов с градиентной заливкой у исходного и конечного объекта должен быть один и тот же тип градиентной заливки. Flash не умеет преобразовывать линейный градиент в радиальный и наоборот.

При применении анимации формы к очень сложному объекту можно добиться лучшего результата, если разбить одну анимацию на несколько и самостоятельно нарисовать промежуточные контуры.

1. К каким объектам можно применить анимацию формы?
2. Какие свойства объектов могут изменяться при анимации формы?
3. Как создать анимацию формы?
4. Каким образом можно использовать импортированные изображения при создании анимации формы?

Упражнения

1 Создайте анимацию формы «Сердце на ладони». Изображение для фона импортируйте.

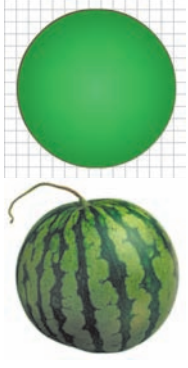
Номер кадра	Изображение
1	
10	

Сохраните анимацию в файле `upr11_1.fla`. Опубликуйте анимацию.

2 Откройте файл. Создайте анимацию формы «Мерцающая звезда».

1. В кадре 1 преобразуйте изображение звезды в обычную графику.
2. В кадр 15 вставьте ключевой кадр.
3. Измените размеры звезды, удерживая клавишу Alt.
4. Создайте анимацию формы.
5. Сохраните изменения в файле `upr11_2.fla` и выполните публикацию.

3 Создайте анимацию формы «Превращение».



1. Нарисуйте в кадре 1 круг с градиентной заливкой.
2. Перейдите в кадр 12 и преобразуйте его в пустой ключевой кадр.
3. Импортируйте в кадр 12 растровое изображение арбуза. Преобразуйте изображение арбуза в векторное и удалите фон (см. пункт 8.3, с. 40—41). Приведите в соответствие размер и место расположения центра круга и арбуза.
4. Кадр 30 преобразуйте в простой кадр.
5. Между ключевыми кадрами создайте анимацию формы.
6. Сохраните результат и выполните публикацию.

4* Дополните результаты выполнения упражнений 1 и 2 анимацией формы так, чтобы получились указанные сюжеты.

1. «Разбитое сердце» — сердце падает и разбивается.
2. «Звездное небо» — несколько мерцающих звезд, период и интенсивность мерцания у каждой свои.

§ 12. Анимация текста

Пример 12.1. Расширяемое текстовое поле.

Пример 12.2. Текстовое поле фиксированной ширины.

Пример 12.3. Текстовый блок.

Пример 12.4. Изменение прозрачности текста.

Пример 12.5. Трансформация текста.

В любой фильм в редакторе Flash может быть добавлен текст. Для добавления текста в фильм используется инструмент **A Текст (Text Tool).**

Место, где будет вводиться текст, определяется щелчком мыши. Появляется выделяющая рамка — текстовое поле. В правом верхнем углу рамки имеется маркер, вид которого определяет тип текстового поля:

- круглый маркер соответствует расширяемому текстовому полю (пример 12.1). Это однострочное текстовое поле, ширина которого автоматически увеличивается при вводе текста;
- прямоугольный маркер соответствует текстовому полю фиксированной ширины (пример 12.2). Если очередной символ не вмещается на текущей строке, в таком поле выполняется автоматический перенос на следующую строку. Ширину поля можно изменить, перетащив маркер.

После завершения ввода образуется текстовый блок с голубой рамкой (пример 12.3).

На **Панели свойств** для текста могут быть установлены: размер, шрифт, стиль, интервал, цвет и способ выравнивания. Можно изменять прозрачность цвета текста (пример 12.4).

Текстовые блоки так же, как и изображения, можно трансформировать: поворачивать, масштабировать, наклонять (пример 12.5). При этом сохраняется возможность редактирования символов текста.

Перед применением операции искажения или изгиба ко всему тексту (пример 12.6) текстовый блок необходимо преобразовать в графический объект. То же самое выполняется для трансформации отдельных символов. Возможны два варианта применения команды **Разбить (Break Apart)**:

1. Однократное применение команды. Происходит разделение текста на символы с сохранением свойств каждого из них как отдельного фрагмента текста. Символы текста заключаются в отдельные рамки. К каждому символу могут быть применены любые операции, допустимые для текстового блока (пример 12.7).

2. Команда применяется дважды. Символы текста преобразуются в графические объекты и выделяются как графические объекты. В этом случае к каждому символу текста можно применять градиентную заливку, искажение и изгиб (пример 12.8).

После применения команды **Разбить** текст уже нельзя редактировать как единый объект.

К текстовым блокам применима только анимация движения (пример 12.9).

Анимацию формы к тексту можно применять только после преобразования

Пример 12.6. Трансформация графического объекта, содержащего текст.

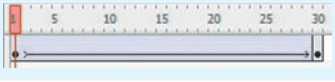
1. Искажение.

2. Изгиб.

Пример 12.7. Изменение свойств каждого символа и применение операций трансформации.

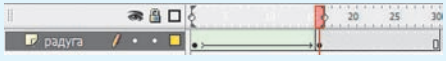
Пример 12.8. Градиентная заливка и изгиб символов.

Пример 12.9. Движение текста с вращением.



Пример 12.10. Применение анимации формы к тексту.

1. В стартовый кадр анимации импортировать изображение и преобразовать его в графику (Ctrl + B).
2. Через несколько кадров вставить пустой ключевой кадр (финишный кадр анимации).
3. В финишный кадр анимации добавить текст и преобразовать его в графику (Ctrl + B дважды).
4. Создать анимацию формы.



символов текста в графику (пример 12.10).

Чтобы символы текста анимировались по отдельности, необходимо:

1. Преобразовать текст в графику.
2. Выполнить команду контекстного меню **Распределить по слоям (Distribute to Layers)**. В результате каждая буква, оставаясь в первом кадре, окажется в своем слое. Слои автоматически получат соответствующие имена.

Если несколько идущих подряд букв должны вести себя одинаково, их можно преобразовать в один графический объект.

1. Какие виды трансформации применимы к текстовому блоку?
2. Для чего текст преобразуют в графический объект?
3. В каком случае сохраняется возможность редактирования текста?
4. Какой вид анимации применим к текстовым блокам?
5. Когда можно применять анимацию к отдельным символам текста?

Упражнения

1 Выполните примеры 12.5—12.10. Сохраните результаты работы.

2 Создайте анимацию по описанию.

1. Установите размеры документа 700 × 200 пикселей.
2. Создайте два слоя — «герб» и «город».
3. В первый кадр слоя «герб» импортируйте изображение герба вашего города (районного или областного центра). При необходимости измените размеры изображения, сохраняя пропорции.
4. В первом кадре слоя «город» создайте текстовый блок с названием города. Установите шрифт, размер и цвет текста в соответствии с образцом.

5. Кадр 19 слоя «город» преобразуйте в ключевой.

6. В стартовом кадре слоя «город» измените размер и место положения текстового блока.

7. В кадры 35 двух слоев добавьте простые кадры.
8. В слое «город» создайте анимацию движения.
9. Сохраните результат и выполните публикацию.

3 Откройте файл upr11_3.fla (результат выполнения упражнения 3 после § 11). Добавьте в фильм анимацию текста.

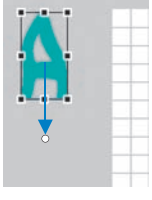
1. Создайте текстовый блок и напишите слово «АРБУЗ» под изображением круга в первом кадре.

2. Разбейте текст на отдельные буквы и распределите их по слоям.

3. В слое «А» преобразуйте выделенную букву в графику. Установите в качестве цвета буквы один из оттенков зеленого и придайте букве необычную форму.

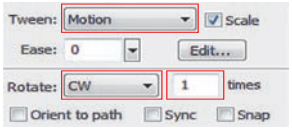
4. Преобразуйте букву в символ для создания анимации движения.
5. Создайте конечную фазу будущей анимации. На некотором удалении по шкале времени вставьте копию первого кадра: 

.

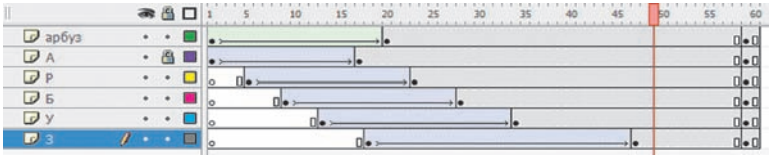


6. Выделите первый кадр в слое «А». Измените стартовые значения анимации. Для этого: вынесите изображение буквы за пределы монтажного стола; измените пропорции буквы и перенесите точку трансформации (центр будущего вращения).

7. Создайте анимацию движения. Буква, постепенно изменяя свои размеры и пропорции, должна вылетать из-за пределов монтажного стола и, совершив оборот, занять свое место.



8. Следуя действиям, описанным в п. 3—7, создайте анимацию движения всех букв. Изменяя место расположения стартовых кадров на шкале времени, добейтесь эффекта появления букв по очереди. Буквы могут двигаться по разным круговым или ломаным траекториям, совершать повороты, изменять свои размеры и пропорции. Время движения и количество поворотов букв тоже может быть разным.

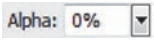


Сохраните изменения в файле `upr12_2.fla` и выполните публикацию.

4 Создайте анимацию «Вода».

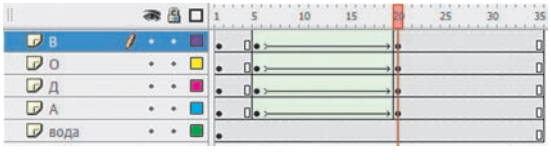
1. Установите размеры документа 400 × 200 пикселей.
2. В первом кадре в центре монтажного стола создайте текстовый блок, как показано на рисунке. Установите шрифт, размер и цвет текста в соответствии с образцом.

3. Разбейте текст на отдельные буквы и распределите их по слоям.
4. В кадр 5 всех четырех слоев добавьте ключевой кадр. Преобразуйте в этом кадре каждую букву в графический объект.
5. В кадр 20 слоев «В», «О» и «Д» вставьте пустой ключевой кадр.
6. Добавьте в каждый слой в кадре 20 текст в соответствии: «В» — «Н», «О» — «2» и «Д» — «О». Уменьшите размер символа «2». Буквы в кадрах 5 и 20 должны иметь приблизительно одинаковое расположение.
7. Кадр 20 слоя «А» сделайте ключевым. Установите прозрачность цвета



.

8. Преобразуйте в кадре 20 каждый символ в графический объект.
9. Создайте анимацию формы (кадр 5 — стартовый, кадр 20 — финишный).
10. Создайте слой «вода» в конце списка слоев. В кадр 1 слоя импортируйте изображение фона.
11. Во всех слоях кадра 35 вставьте простой кадр:



12. Сохраните результат и выполните публикацию.

Глава 3
ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

§ 13. Основные алгоритмические конструкции

13.1. Алгоритм и алгоритмические конструкции

В 7-м классе вы познакомились с основными алгоритмическими конструкциями. Для решения задач по программированию были выделены основные этапы (пример 13.1).

Алгоритм — конечная последовательность точных действий, формальное выполнение которых позволяет получить решение задачи для любого допустимого набора исходных данных.

Исполнитель — человек, группа людей или техническое устройство, которые способны правильно выполнять команды алгоритмов. В дальнейшем будем рассматривать только исполнителя-устройство с ограниченным набором команд. Набор команд одного исполнителя называют **системой команд исполнителя**. Команды компьютерного исполнителя могут быть реализованы в виде процедур и функций.

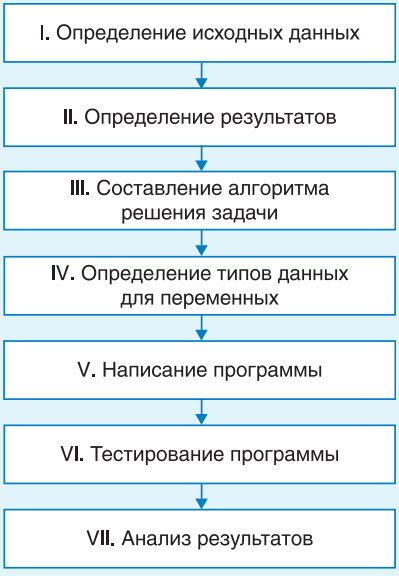
Все команды исполнителя делят на группы:

1. Команды, которые непосредственно выполняет исполнитель.

2. Команды, изменяющие порядок выполнения других команд исполнителя.

Любой алгоритм может быть записан с использованием трех базовых алгоритмических конструкций:

Пример 13.1. Этапы решения задачи по программированию:



В процессе решения задачи некоторые этапы приходится повторять до тех пор, пока анализ результатов не покажет, что задача решена верно.

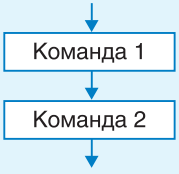
Для поиска ошибок можно использовать средства отладки программ (см. *Приложение 3*, с. 161—162).

В 1966 г. итальянские математики Коррадо Бём (1923—2017) и Джузеппе Джакопини (1936—2001) сформулировали и доказали положение структурного программирования, согласно которому любой исполняемый алгоритм может быть преобразован к структурированному виду, т. е. виду, когда ход выполнения алгоритма определяется при помощи трех структур управления: последовательной, ветвлений и циклов.

Полностью концепция структурного программирования была разработана в середине 70-х гг. при участии Э. Дейкстры.

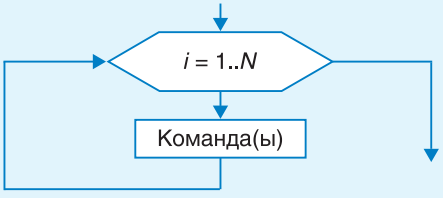
Пример 13.2. Блок-схемы алгоритмических конструкций:

1. Следование:

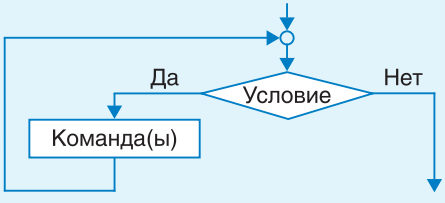


2. Цикл:

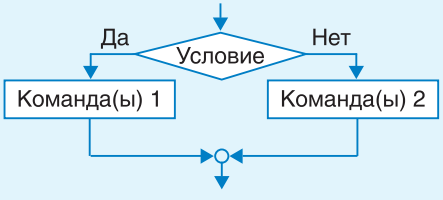
1) цикл с параметром (значение параметра изменяется от 1 до N)



2) цикл с предусловием



3. Команда ветвления



следование, цикл и **ветвление** (пример 13.2).

Команды цикла и ветвления управляют порядком выполнения других команд в программе и относятся к **командам управления (управляющим конструкциям).**

Последовательность команд, исполнителем которой является компьютер, называется **программой.** Программа представляет собой запись на некотором формальном языке — языке программирования. Командами в языке программирования считают:

- операторы (оператор присваивания, оператор ветвления, оператор цикла и др.);
- вызовы вспомогательных алгоритмов (как встроенных в библиотеки, так и созданных пользователем).

13.2. Алгоритмическая конструкция *следование*

Алгоритмическая конструкция ***следование*** — последовательность команд алгоритма, которые выполняются в том порядке, в котором они записаны. Среди команд, образующих алгоритмическую конструкцию *следование*, отсутствуют команды, меняющие порядок выполнения других команд.

В 7-м классе, изучая язык Pascal, вы использовали следующие команды (пример 13.3):

- процедуры для ввода и вывода данных;
- оператор присваивания.

Для **ввода данных** предназначена команда `read()`. В скобках через запятую перечисляются имена переменных, значения которых необходимо ввести.

Для **вывода данных** используют команду `write()`. Она позволяет выводить текстовые сообщения и числовые значения. Текстовые сообщения записываются в кавычках, выводятся в виде последовательности символов так, как записаны, и не анализируются при выполнении.

При использовании команды `writeln()` после вывода сообщения или числа происходит перевод курсора на следующую строку.

Оператор присваивания предназначен для того, чтобы:

- задавать значения переменным;
- вычислять значение выражения (результат будет записан как значение переменной).

Формат записи оператора присваивания:

<имя переменной> := <выражение>;

В записи арифметического выражения используются знаки математических действий: сложения (+), вычитания (–), умножения (*), деления (/), а также целочисленного деления (div) и нахождения остатка (mod). Следует помнить, что операция деления (/) используется при вычислениях с данными типа `real`. Для данных типа `integer` используются операции `div` и `mod`.

Нередко в одной программе приходится выполнять одну и ту же последовательность команд несколько раз. В этом случае удобно использовать

Пример 13.3. Даны x, y. Написать программу для вычисления значения выражения

$$a = \frac{2x}{7 + y^2}(x - y).$$

Этапы выполнения задания

I. Определение исходных данных: переменные x, y.

II. Определение результатов: переменная a.

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Вычисление значения выражения.
3. Вывод результата.

IV. Описание переменных: все переменные, определенные для решения задачи, имеют тип `real`.

V. Программа:

```
var x,y,a: real;
begin
  write('Введите x =');
  read(x);
  write('Введите y =');
  read(y);
  a:= 2 * x * (x-y)/(7 + y * y);
  writeln('a =',a);
End.
```

VI. Тестирование программы:

Запустить программу и ввести значения $x = 3.8$, $y = 2.7$. Результат:

Окно вывода

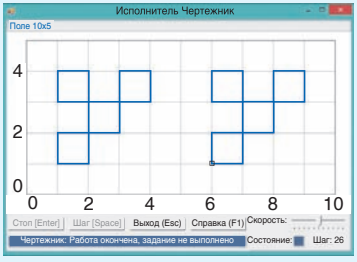
```
Введите x = 3.8
Введите y = 2.7
a = 0.585024492652204
```

VII. Правильность вычислений проверить на калькуляторе.

В примере алгоритмическая конструкция *следование* образована командами:

- вывод сообщений;
- ввод значений переменных;
- команда присваивания;
- вывод результата.

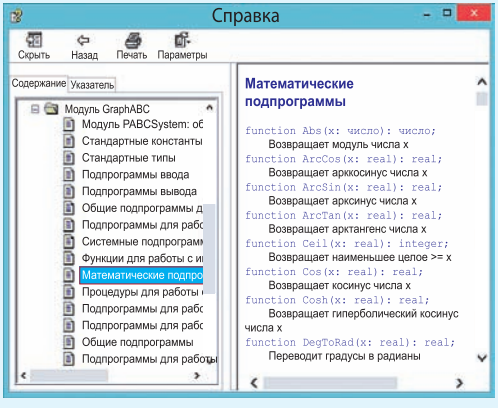
Пример 13.4. Написать программу для вывода изображения:



Изображение состоит из двух одинаковых фигур. Оформим вспомогательный алгоритм `Figura` для изображения одной фигуры. Программа:

```
uses Drawman;
procedure Figura;
begin
  PenDown; OnVector(1, 0);
  OnVector(0, 3); OnVector(-1, 0);
  OnVector(0, -1); OnVector(3, 0);
  OnVector(0, 1); OnVector(-1, 0);
  OnVector(0, -2); OnVector(-2, 0);
  OnVector(0, -1); PenUp;
end;
begin
  Field(10, 5);
  ToPoint(1, 1); Figura;
  ToPoint(6, 1); Figura;
end.
```

Пример 13.5. Перечень математических функций можно посмотреть в справке PascalABC:



вспомогательный алгоритм, который можно выполнять нужное число раз, обращаясь к его названию.

Вспомогательный алгоритм — алгоритм, который можно использовать в других алгоритмах, указав его имя и, если необходимо, значения параметров.

Вспомогательный алгоритм решает некоторую часть основной задачи. Вызов вспомогательного алгоритма является командой, которая может заменять несколько команд.

Вспомогательные алгоритмы вы использовали при написании программ для учебных компьютерных исполнителей Чертежник и Робот (пример 13.4). Команды `read` и `write` тоже реализованы как вспомогательные алгоритмы.

При вычислениях часто используются различные математические функции (пример 13.5). Эти функции реализованы как встроенные вспомогательные алгоритмы и могут применяться при записи арифметических выражений. Аргументы функций всегда записываются в скобках. Некоторые из функций приведены в таблице (другие можно посмотреть в *Приложении 3*, с. 158).

Запись на языке Pascal	Описание
`abs(x)`	Находит модуль числа x
`sqr(x)`	Возводит число x в квадрат
`sqrt(x)`	Находит корень квадратный из числа x. Результат — всегда число типа `real`

Запись на языке Pascal	Описание
`trunc(x)`	Находит целую часть действительного числа x (`real`). Результат — число типа `integer`
`frac(x)`	Находит дробную часть действительного числа x (`real`). Результат — число типа `real`
`sin(x)`	Вычисляет синус числа x. Число x задается в радианах[1]
`cos(x)`	Вычисляет косинус числа x. Число x задается в радианах
`RadToDeg(x)`	Переводит радианы в градусы
`DegToRad(x)`	Переводит градусы в радианы

Аргументом функции может быть число, переменная, выражение или другая функция: `sin(DegToRad(45))`, `sqrt(abs(-16))`.

В примере 13.6 используются математические функции для возведения числа в квадрат и вычисления квадратного корня.

Пример 13.6. Задана длина стороны квадрата a. Написать программу нахождения площади квадрата и длины его диагонали.

Этапы выполнения задания

I. Исходные данные: длина стороны, переменная a.

II. Результат: переменные S (площадь) и d (длина диагонали).

III. Алгоритм решения задачи.

1. Ввод исходных данных.
2. Вычисление площади по формуле $S = a^2$.
3. Вычисление длины диагонали по формуле $d = a\sqrt{2}$.
4. Вывод результата.

IV. Описание переменных: все переменные, определенные для решения задачи, имеют тип `real`.

V. Программа:

```
var a, S, d: real;
begin
 write('Введите a ='); read(a);
 S:= sqr(a); d:= a*sqrt(2);
 writeln('S =',s); writeln('d =',d);
end.
```

VI. Тестирование программы.

Запустить программу и ввести значение $a = 5.6$. Результат:

Окно вывода

```
Введите a = 5.6
S = 31.36
d = 7.91959594928933
```

Правильность вычислений можно проверить на калькуляторе.

?

1. Что такое алгоритм?
2. Перечислите основные алгоритмические конструкции.
3. Какая команда используется в языке Pascal для ввода данных?
4. Какие команды используются в языке Pascal для вывода данных?
5. Для чего нужна команда присваивания?
6. В каких случаях удобно использовать вспомогательный алгоритм?
7. Какие математические функции могут использоваться при записи арифметических выражений?

[1] Радиан — единица измерения углов в Международной системе единиц (1 радиан соответствует величине развернутого угла 180°).

Упражнения

1 Расставьте команды программы в правильном порядке так, чтобы можно было вычислить значение выражения $a = \frac{2x}{x^2 + 4}$.

```
1. writeln('a = ',a);
2. write('Введите значение x = ');
3. End.
4. Var x,y,a: real;
5. Begin
6. a: = 2 * x/(x * x+4);
7. read(x);
```

2 Определите типы данных для каждой переменной, использованной в операторе присваивания.

```
1. y: = sqrt(a – 4)/16;
2. z: = sqr(3 * a + 2);
3. a: = abs(a – 4.2);
4. d: = x mod 2;
5. y: = int(a);
6. y: = trunc(a);
7. y: = frac(a);
8. s: = sin(3.14 * r).
```

3 Найдите и исправьте ошибки в программах.

1.
```
var x, y, z1, z2:integer;
begin
 write('Введите x =');
 read(x);
 write('Введите y =');
 read(y);
 z1:= int(x/y);
 z2:= frac(x/y);
 write('Целая часть =',z1);
 write('Дробная часть =',z2);
end.
```

2.
```
var x, y, z1, z2:real;
begin
 write('Введите x  =');
 read(x);
 write('Введите y  =');
 read(y);
 z1:=x div y;
 z2:=x mod y;
 write('Целая часть =',z1);
 write('Остаток =',z2);
end.
```

4 Даны x и z. Измените программу из примера 13.4 так, чтобы вычислялось значение выражения $a = \frac{2x}{\sqrt{z^2 + 9}}$.

5 Заданы три числа. Напишите программу для нахождения среднего арифметического этих чисел.

6 Заданы два числа. Напишите программу для нахождения частного от деления первого числа на второе и округлите результат до ближайшего целого.

7 Даны гипотенуза и катет прямоугольного треугольника. Напишите программу для нахождения второго катета и площади треугольника.

8* Заданы два числа. Напишите программу для нахождения частного этих чисел. Округлите результат до десятых, оставив в дробной части одну цифру.

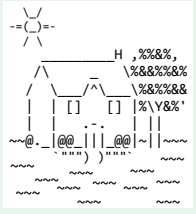
§ 14. Графические возможности среды программирования PascalABC

14.1. Основы работы с графикой

Вы уже знакомы с графическими редакторами, в которых для построения изображений на компьютере используются графические примитивы — простые геометрические фигуры: прямоугольник, окружность, эллипс, отрезок и т. д. Графический редактор — программа, написанная на каком-либо языке программирования.

Для работы с графикой языки программирования используют специальные библиотеки (модули), содержащие наборы команд для построения изображений. В PascalABC для работы с графикой используется библиотека GraphABC. Для подключения этой библиотеки в программе записывается команда **uses** `GraphABC;`

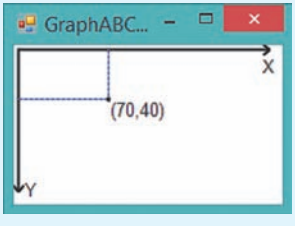
Положение фигур задается координатами в графическом окне. Координатная плоскость в нем отличается от той, которую вы используете на уроках математики. Началом координат является верхний левый угол графического окна — точка (0; 0) (пример 14.1). Координаты задают порядковый номер пикселя по горизонтали и вертикали, поэтому они могут быть только целыми числами. Отсчет значений координаты x происходит слева направо, а координаты y — сверху вниз. По умолчанию создается графическое окно размером 640 × 480 пикселей.



Первые компьютеры не имели возможностей работы с графикой. На печатающих устройствах выводились «картинки», состоящие из символов[1].

В 1958 г. был запущен компьютер Lincoln TX-2, впервые использующий графический экран. В 1981 г. начали применять цвета. В графическом режиме при разрешении 320 × 200 пикселей использовались 4 цвета из стандартных палитр: пурпурный, сине-зеленый, белый, черный или красный, зеленый, желтый, черный.

Пример 14.1. Графическое окно среды программирования PascalABC с изображением координатных осей и точки с координатами (70, 40).

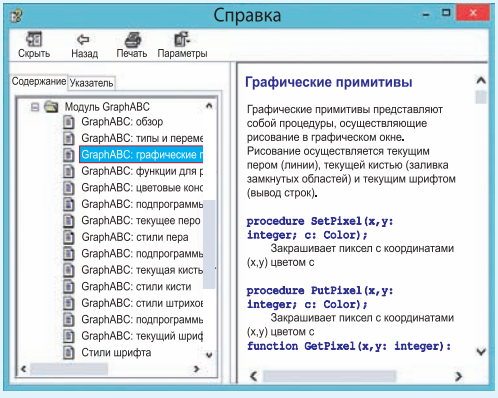


Точка расположена на расстоянии 70 пикселей от левого края окна и на расстоянии 40 пикселей от верхнего края.

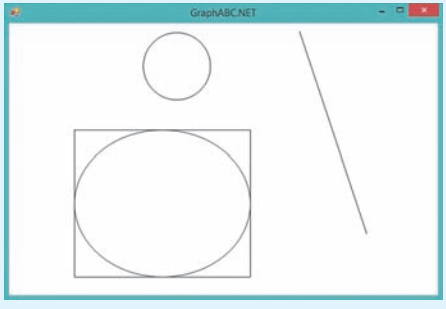
Размеры графического окна можно задать командой `SetWindowSize(n,m);` В скобках указаны размеры окна по горизонтали и вертикали.

[1] https://www.asciiart.eu/buildings-and-places/houses (дата доступа: 26.07.2018).

Пример 14.2. Работа со справочной системой. Для перехода в справочник необходимо нажать клавишу F1 или выполнить команду меню: **Помощь** → → **Справка**. В открывшемся окне перейти в раздел **Стандартные модули** и выбрать **Модуль GraphABC**.



Пример 14.3. Графические примитивы:



Программа для их рисования:

```
uses GraphABC;
begin
  //Круг
  Circle(250, 125, 30);
 //Прямоугольник
 Rectangle(100,200,400,450);
  //Эллипс
 Ellipse(100,200,400,450);
  //Отрезок
 Line(450, 50, 550, 350);
end.
```

14.2. Работа со справочной системой среды программирования PascalABC

В библиотеке GraphABC содержится большое количество команд. Эти команды описаны в справочной системе среды PascalABC (пример 14.2). Здесь есть описание графических примитивов, названия цветовых констант, описание работы с пером и кистью, команды работы с графическим окном.

Команды библиотеки GraphABC — вспомогательные алгоритмы, записанные как отдельные процедуры. Использование команды в программе означает вызов соответствующего алгоритма.

14.3. Основные графические примитивы

Рассмотрим команды для рисования графических примитивов:

- `Line(x1,y1,x2,y2)` — отрезок, соединяющий точки с координатами ($x1$, $y1$) и ($x2$, $y2$).
- `MoveTo(x,y)` — устанавливает текущую позицию рисования в точку (x, y);
- `LineTo(x,y)` — отрезок от текущей позиции до точки (x, y);
- `Rectangle(x1,y1,x2,y2)` — прямоугольник, заданный координатами противоположных вершин ($x1$, $y1$) и ($x2$, $y2$);
- `Circle(x1,y1,r)` — круг с центром в точке ($x1$, $y1$) и радиусом r;
- `Ellipse(x1,y1,x2,y2)` — овал (эллипс), вписанный в прямоугольник с координатами противоположных вершин ($x1$, $y1$) и ($x2$, $y2$).

(Рассмотрите пример 14.3.)

Команды для рисования других графических примитивов имеются в справочной системе и в *Приложении 3* (с. 159).

Пример 14.4. Написать программу, которая строит изображение домика,

используя процедуры Line, Lineto, Rectangle, Circle.

Этапы выполнения задания

I. Исходные данные: результат работы программы не зависит от исходных данных.

II. Результат: готовый рисунок.

III. Алгоритм решения задачи.

Рисунок состоит из: прямоугольников, отрезков, круга. Для расчета координат рекомендуется предварительно сделать рисунок на листе бумаги в клеточку.

IV. Описание переменных. Переменные не используются.

14.4. Работа с пером и кистью

В графических редакторах, прежде чем рисовать какие-либо фигуры, устанавливают их цвет. Обычно выбирают два цвета. Цвет 1 определяет цвет линий и контуров фигур, Цвет 2 используется для заливки фигур. Кроме того, можно изменять стиль линий и заливки, а также определять толщину линий.

В графическом режиме PascalABC настройки линии определяет перо (Pen), а настройки внутренней области фигур — кисть (Brush). Команды для работы с кистью и пером приведены в таблице.

Команда	Описание
SetPenColor	Цвет линий
SetPenWidth	Толщина линии
SetPenStyle	Стиль линий
SetBrushColor	Цвет заливки
SetBrushStyle	Стиль заливки
SetBrushHatch	Вид штриховки для заливки

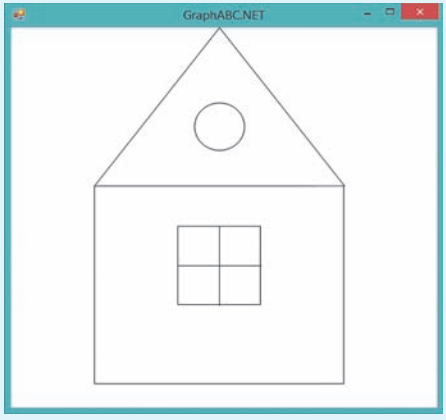
Пример 14.4.

V. Программа:

```
uses GraphABC;
begin
  //Дом
  Rectangle(100,200,400,450);
  //Окно
  Rectangle(200,250,300,350);
  Line(250, 250, 250, 350);
  Line(200, 300, 300, 300);
  //Крыша
  MoveTo(100,200);
  LineTo(250, 0);
  LineTo(400, 200);
  Circle(250, 125, 30);
end.
```

VI. Тестирование программы:

Запустить программу. Результат:

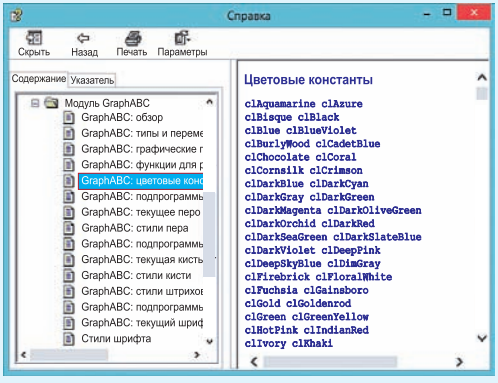


Обратите внимание, что при написании команд у вас появляются подсказки:

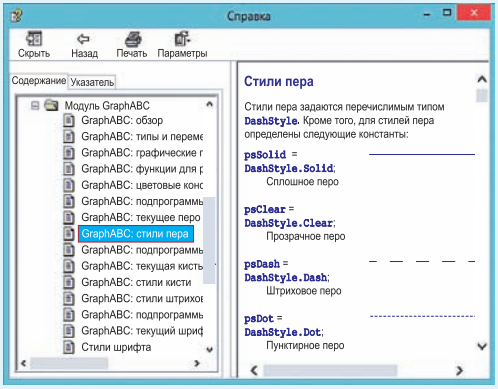
Подсказка появляется также при наведении указателя мыши на уже написанную команду.

Если установить текстовый курсор на команду и нажать F1, то откроется страница из справочника с описанием этой команды.

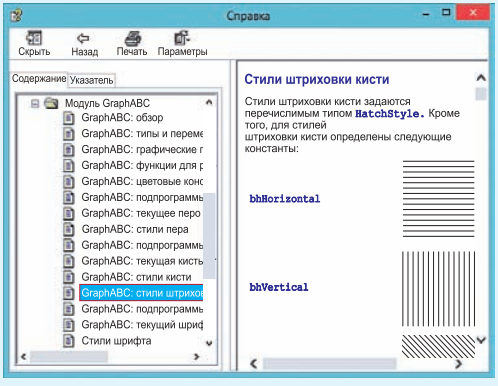
Пример 14.5. Цветовые константы.



Пример 14.6. Стили пера.



Пример 14.7. Стили штриховки кисти.



Значение, которое нужно установить для каждой из команд, записывается в скобках. Например:

- `SetPenColor(clRed)` — красный цвет рисования линий;
- `SetBrushColor(clBlue)` — синий цвет заливки фигур;
- `SetPenWidth(3)` — толщина пера в 3 пикселя;
- `SetPenStyle(psDot)` — штриховая линия;
- `SetBrushStyle(bsHatch)` — штриховая заливка;
- `SetBrushHatch(bhCross)` — штриховка в клеточку.

Значения цветовых констант, стилей линий и заливок можно найти в справочной системе среды PascalABC (примеры 14.5—14.7) и в *Приложении 3* (с. 160).

Команды для установки цвета и стиля записывают перед командой рисования фигуры. Эти команды действуют до тех пор, пока цвет или стиль не будет изменен. Если, например, для пера была установлена толщина в 3 пикселя, то отрезки и границы фигур будут иметь толщину в 3 пикселя до новой смены толщины пера.

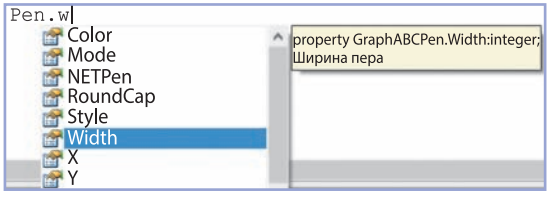
Для фигур по умолчанию установлена заливка белым цветом. Если цвет кисти выбрать до рисования фигуры, то фигура будет закрашена установленным цветом. Цвет уже нарисованной фигуры можно изменить с помощью команды заливки:

`FloodFill(x,y,c);` — заливает ограниченную область одного цвета цветом `c`, начиная с точки внутри области `(x,y)` (в примере 14.8 показано, как раскрашен домик из примера 14.4).

Среда программирования PascalABC позволяет обращаться к свойствам кисти и пера по-другому. Так, например, для изменения цвета (стиля, толщины) можно записать

```
Pen.Color := clRed;
Pen.Style := psDot;
Pen.Width := 3;
```

Подсказка среды выглядит следующим образом:



Вторую часть команды можно выбрать из выпадающего списка.

При изучении векторной графики вы познакомились с цветовой моделью RGB, которая позволяет записать любой цвет тремя составляющими: красной, зеленой и синей. Функция `RGB (r, g, b)` позволяет определить цвет по трем составляющим. Так, команда `SetPenColor (clRed);` аналогична команде `SetPenColor (RGB(255,0,0));` Такой способ задания цвета позволяет задавать цвета, значения которых не описаны цветовыми константами. Значения составляющих цвета можно посмотреть в графическом редакторе Paint (пример 14.9).

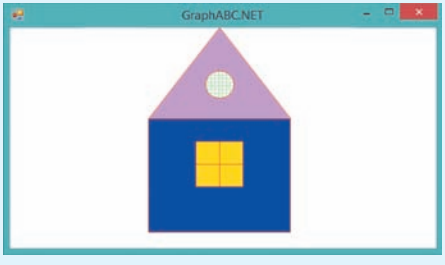
В графическом режиме PascalABC можно выводить в графическое окно тексты и числа.

`TextOut(x,y,z);` — выводит строку или число `z` в прямоугольник с координатами левого верхнего угла `(x,y)`.

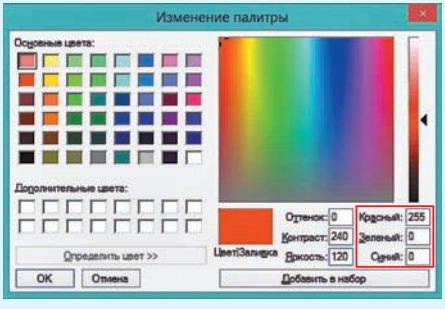
Пример 14.8. Программа:

```
uses GraphABC;
begin
  //Дом
  SetPenColor(RGB(255,0,0));
  SetBrushColor(clBlue);
  Rectangle(100,200,400,450);
  //Окно
  SetBrushColor(clYellow);
  Rectangle(200,250,300,350);
  SetPenColor(clRed);
  SetPenStyle(psDot);
  SetPenWidth(2);
  Line(250,250,250,350);
  Line(200,300,300,300);
  //Крыша
  SetPenStyle(psSolid);
  SetPenWidth(1);
  Line(100, 200, 250, 0);
  Line(250, 0, 400, 200);
  SetBrushStyle(bsHatch);
  SetBrushColor(clLightGreen);
  SetBrushHatch(bhCross);
  Circle(250, 125, 30);
  FloodFill(250,70, clPlum);
end.
```

Результат работы программы:



Пример 14.9. Составляющие цвета:

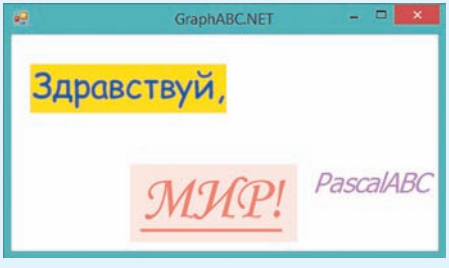


Пример 14.10. Выведем в графическое окно приветствие миру, используя разные свойства текста.

Программа:

```
uses GraphABC;
begin
 //Цвет фона для текста
 SetBrushColor(clYellow);
 SetFontName('Comic Sans MS');
 //Цвет букв
 SetFontColor(clBlue);
 //Размер шрифта
 SetFontSize(25);
 //Полужирный шрифт
 SetFontStyle(fsBold);
 TextOut(20,30,'Здравствуй,');
 SetBrushColor(clPink);
 StFontName('Monotype Corsiva');
 SetFontColor(clSalmon);
 SetFontSize(50);
 SetFontStyle(fsUnderline);
 TextOut(120,130,'МИР!');
 //Прозрачный фон
 SetBrushStyle(bsClear);
 SetFontName('Tahoma');
 SetFontColor(clViolet);
 SetFontSize(20);
 SetFontStyle(fsItalic);
 TextOut(300,130,'PascalABC');
end.
```

Результат выполнения программы:



Если нужно вывести строку, то ее символы заключают в кавычки, для вывода числа можно использовать переменные или значения чисел. Если в качестве *z* записать арифметическое выражение, то будет выведено его значение.

Для изменения параметров текста применяют следующие команды:

Команда	Описание
SetFontColor	Цвет символов
SetFontSize	Размер символов
SetFontName	Имя текущего шрифта
SetFontStyle	Стиль текста

Команда SetBrushColor устанавливает цвет прямоугольника, внутри которого будет находиться текст (пример 14.10). Команда действует до тех пор, пока цвет не будет изменен. Для записи текста без фона нужно установить прозрачный фон кисти: SetBrushStyle(bsClear).

Все параметры для текста задаются до команды вывода его в графическое окно. Имя шрифта, которым будет выведен текст, заключается в кавычки. Возможные варианты можно посмотреть в Word. Стиль текста может иметь следующие значения: fsNormal (обычный), fsBold (жирный), fsItalic (наклонный), fsUnderline (подчеркнутый) или их комбинации: fsBoldUnderline (жирный подчеркнутый).

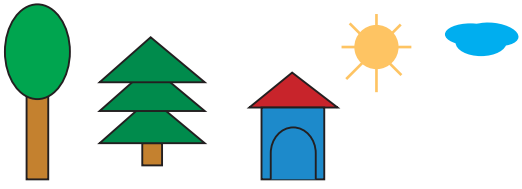
1. Какая библиотека используется для подключения графики в PascalABC?
2. Как определена система координат в графическом окне?
3. Как задать размеры графического окна?
4. Как найти описание графических примитивов в справочнике?
5. Как можно изменить цвет линий, заливки?
6. Как вывести текст в графическом окне?

Упражнения

1 Подпишите изображение из примера 14.8.

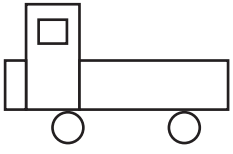
2 Дополните изображение домика из примера 14.8 изображениями трубы и дыма из трубы в виде нескольких овалов:

3 Дополните результат, полученный при выполнении задания 2, какими-либо из предложенных изображений или придумайте свои.

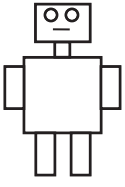


4 Напишите программу для создания изображения. Раскрасьте данное изображение по своему усмотрению. Дополнительные команды для построения графических примитивов можно найти в справочной системе.

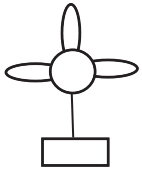
Грузовик
1



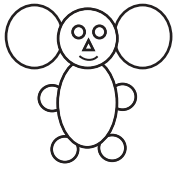
Робот
2



Цветок
3



Чебурашка
4



§ 15. Простые и составные условия

15.1. Логический тип данных

Напомним изученные в 7-м классе понятия *высказывание* и *условие для исполнителя*.

Высказывание — повествовательное предложение (утверждение), о котором можно сказать, истинно оно или ложно.

Условием для исполнителя является известное ему высказывание, которое может соблюдаться (быть

Тип `Boolean` назван в честь английского математика и логика Джорджа Буля, занимавшегося вопросами математической логики в XIX в.

Данный тип присутствует в подавляющем большинстве языков программирования. В некоторых языках реализуется через числовой тип данных. Тогда за значение ложь принимается 0, а за значение истина — 1.

Пример 15.1. Примеры логических выражений:

- `3 < 7` — логическое выражение, значение которого `true`;
- `2 + 2 * 2 = 8` — логическое выражение, значение которого `false`;
- `abs(-5) > abs(3)` — логическое выражение, значение которого `true`;
- `y >= sqr(x)` — логическое выражение, значение которого можно определить, только зная значения переменных x и y. При $x = 2$ и $y = 10$ значение выражения — `true`. При $x = 10$ и $y = 2$ — `false`.

Проверим истинность этих выражений в программе:

```
var a1, a2, a3, a4, a5: boolean;
  x, y: integer;
begin
  a1 := 3 < 7;
  writeln('a1 =', a1);
  a2 := 2 + 2 * 2 = 8;
  writeln('a2=', a2);
  a3 := abs(-5) > abs(3);
  writeln('a3 =', a3);
  x := 2; y := 10;
  a4:= y > = sqr(x);
  writeln('a4 = ', a4);
  x := 10; y := 2;
  a5 := y >= sqr(x);
  writeln('a5  =', a5);
end.
```

Результат работы программы:

Окно вывода

```
a1 = True
a2 = False
a3 = True
a4 = True
a5 = False
```

По умолчанию `false < true`.

истинным) либо не соблюдаться (быть ложным).

В языке программирования Pascal для работы с условиями определен **логический тип данных boolean**. Величины типа `boolean` могут принимать два значения — `false` (ложь) и `true` (истина).

Значения `false` и `true` получаются в результате выполнения операций сравнения над числовыми данными. Для сравнения используют знаки, указанные в таблице.

Операция	PascalABC
Равно (=)	=
Не равно (≠)	<>
Больше (>)	>
Меньше (<)	<
Больше или равно (⩾)	>=
Меньше или равно (⩽)	<=

Сравнивать можно константы, переменные, арифметические и логические выражения.

Логическое выражение — выражение, принимающее одно из двух значений: `true` или `false`.

Логические выражения можно присваивать переменным типа `boolean`, а также выводить их значения на экран: будет выведено слово `false` или `true` соответственно (пример 15.1). Условия для исполнителя являются частным случаем логических выражений.

Пример 15.2. Написать программу, которая выведет на экран значение `true` или `false` в зависимости от того, является ли введенное число x четным или нет.

Этапы выполнения задания

I. Исходные данные: x (введенное число).

II. Результат: a (true или false).

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Вычисление значения логической переменной. Число является четным, если остаток от деления его на 2 равен нулю. Значение переменной a определяется значением выражения x **mod** 2 = 0.

3. Вывод результата.

IV. Описание переменных: x — integer, a — boolean.

Пример 15.2.

V. Программа:

```
var x: integer;
  a: boolean;
begin
  write('Введите x =');
  read(x);
  a := x mod 2 = 0;
  write('Число четное – ',a);
end.
```

VI. Тестирование

Запустить программу и ввести значение x = 6. Результат:

Окно вывода

```
Введите x = 6
Число четное - True
```

Запустить программу и ввести значения x = 11. Результат:

Окно вывода

```
Введите x = 11
Число четное - False
```

15.2. Составные условия

С высказываниями можно производить логические операции (**НЕ, И, ИЛИ**). Для логических переменных также определены логические операции, соответствующие операциям над высказываниями: **not**, **and**, **or**.

Логические выражения, в которых наряду с простыми условиями (сравнениями) используются логические операции, называют **составными условиями**.

Приведем таблицы истинности логических операций.

Логическая переменная		Результат операции		
A	*B*	**not** *A*	*A* **and** *B*	*A* **or** *B*
True	True	False	True	True
False	True	True	False	True
True	False	False	False	True
False	False	True	False	False

В языке PascalABC реализована логическая операция **xor** — исключающее **ИЛИ**. Этой операции соответствует высказывание: «Только одно из двух высказываний может быть истинно». Таблица истинности для операции **xor**:

A	*B*	*A* **xor** *B*
True	True	False
False	True	True
True	False	True
False	False	False

Все логические операции могут применяться к числам типа integer. Число рассматривается в двоичном представлении, и операции применяются к битам числа. Бит, равный 1, представляется как истина, а бит, равный нулю, — как ложь.

Пример 15.3. Определение порядка действий для выражения (*a*, *d* — boolean, *c*, *b* — integer):

a **or** (c < b) **and** d

Первым выполняется сравнение *c* и *b*, затем логическая операция **and**, потом — **or**.

Пример 15.4*. Рассмотрим выражение:

a < b **and** c < d

Если *a*, *b*, *c*, *d* имеет тип integer, то получим ошибку: «Операция '<' не применима к типам boolean и integer» (с помощью знака '<' нельзя сравнивать число и логическую переменную). Если переменные имеют тип real, то возникнет ошибка: «Операция '**and**' не применима к типу real». Правильная запись выражения:

(a < b) **and** (c < d)

Все вышеперечисленные ошибки возникают потому, что операция **and** обладает большим приоритетом по отношению к операциям <. Поэтому сначала будет производиться попытка выполнить операцию b **and** c, а затем сравнения.

Пример 15.5. Построение отрицаний:

not not a = a;
not (a **and** b) = (**not** a) **or** (**not** b);
not (a **or** b) = (**not** a) **and** (**not** b).

Рассмотрим выражение **not** a < b с переменными *a* и *b* типа integer. Здесь операция **not** относится к переменной *a*, поэтому в двоичном представлении числа *a* биты, равные 1, будут заменены на 0, а биты, равные 0, — на 1. Затем полученный результат сравнится с числом *b*. Для отрицания сравнения выражение нужно записать так: **not** (a < b).

В логических выражениях могут встречаться как арифметические операции, так и логические. Порядок выполнения операций определяется их приоритетом:

1) **not**;
2) *, /, **div**, **mod**, **and**;
3) +, –, **or**;
4) =, <>, <, >, <=, >=.

(Рассмотрите пример 15.3.)

Операции с одинаковым приоритетом выполняются по порядку, слева направо. Для изменения порядка выполнения операций применяют скобки (пример 15.4).

При составлении программ часто нужно строить отрицания сложным логическим выражениям. Для этого полезно использовать тождества, известные из алгебры логики (пример 15.5), и следующую таблицу:

Условие	Противоположное условие (отрицание условия)
$a < b$	$a >= b$
$a > b$	$a <= b$
$a = b$	$a <> b$

Пример 15.6. Написать программу, которая выдаст на экран значение true или false в зависимости от того, находится ли число *B* между числами *A* и *C*.

Этапы выполнения задания

I. Исходные данные: переменные *A*, *B*, *C* (вводимые числа).

II. Результат: rez (True или False).

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Вычисление значения логической переменной. Рассмотрим два случая.

Верно неравенство: $A < B < C$. Этому неравенству соответствует логическое выражение: (A < B) **and** (B < C). Присвоим переменной r1 значение этого выражения.

Верно неравенство: $A > B > C$. Этому неравенству соответствует логическое выражение: (A > B) **and** (B > C). Присвоим переменной r2 значение этого выражения.

Ответом на задачу будет значение логического выражения r1 **or** r2.

3. Вывод результата.

IV. Описание переменных: A, B, C — integer, r1, r2, rez — boolean.

Для работы с логическими величинами могут использоваться функции. Функция Ord (порядковый номер значения) позволяет преобразовать логическое значение в числовое: Ord(false) = 0, а Ord(true) = 1. Функции Pred (предшествующее значение) и Succ (последующее значение) позволяют преобразовывать логические значения:

```
D := Pred(true);  {D = false}
E := Succ(false);  {E = true}
```

Пример 15.6.

V. Программа:

```
var A, B, C: integer;
 r1, r2, rez: boolean;
begin
 writeln('Введите A, B, C');
 read(A, B, C);
 r1 := (A < B) and (B < C);
 r2 := (A > B) and (B > C);
 rez := r1 or r2;
 write('Число B между числами
       A и C – ',rez);
end.
```

VI. Тестирование.

Запустить программу и ввести значения A = 5, B = 0, C = –5. Результат:

Окно вывода

```
Введите A, B, C
5 0 -5
Число B между числами A и C - True
```

Запустить программу и ввести значения A = –2, B = –7, C = 5. Результат:

Окно вывода

```
Введите A, B, C
-2 -7 5
Число B между числами A и C - False
```

VII. Анализ результатов. Для полного тестирования программы нужно проверить все возможные случаи взаимного расположения A, B, C (их всего 6).

1. Что такое составное условие?
2. Назовите логические операции, используемые в PascalABC.
3. Какой приоритет у логической операции **not** (**and**, **or**)?

Упражнения

1 Сформулируйте и реализуйте обратную задачу для примера 15.2: для всех тех случаев, для которых в исходной задаче было true, нужно вывести false и, наоборот, для всех тех случаев, в которых в исходной задаче получалось false, получить true.

2 В PascalABC определена логическая функция odd(x). Значение этой функции true, если число x является нечетным, и false, если x — четное. Измените программу примера 15.2, используя функцию odd.

3 Определите, что делают следующие программы, и дополните команду вывода.

1.
```
var x: integer;
    a: boolean;
begin
  write('Введите x =');
  read(x);
  a := x mod 10 = 0;
  write('Число ... – ',a);
end.
```

2.
```
var x: integer;
    a: boolean;
begin
  write('Введите x  =');
  read(x);
  a := (x>10) and (x < 100);
  write('Число ... – ',a);
end.
```

4 Напишите программу, которая выведет на экран значение `true` или `false`, в зависимости от того, является ли введенное число *x* положительным или нет.

5 Напишите программу, которая выведет на экран значение `true` или `false`, в зависимости от того, является ли введенное число *x* четырехзначным или нет.

6* Заданы два положительных числа *x* и *y*. Определите, верно ли, что первое число меньше второго и хотя бы одно из них нечетное. Выведите на экран `true` или `false`.

§ 16. Оператор ветвления

Использование управляющих конструкций предполагает запись программы в структурированном виде. Структурированность программ достигается за счет отступов, регулирующих размещение вложенных алгоритмических конструкций.

Можно соблюдать следующее правило: при движении курсора вниз от «начала» структуры до ее «конца» на пути курсора могут встретиться только пробелы. Все, что находится «внутри» структуры, размещается правее.

Кнопка  позволяет преобразовать код программы к структурированному виду.

Пример 16.1.
V. Программа:
```
var x: integer;
begin
 write('Введите x = '); read(x);
 if x > 0 then
   write('положительное')
 else
   write('не положительное');
end.
```

16.1. Запись оператора ветвления

Алгоритмическая конструкция *ветвление* (см. блок-схему в примере 13.2, с. 60) обеспечивает выполнение одной или другой последовательности команд в зависимости от истинности или ложности некоторого условия.

Оператор ветвления — команда, реализующая алгоритмическую конструкцию *ветвление* на языке программирования.

Для записи оператора ветвления используют команды **if**. Формат команды:
```
if <условие> then
begin
  Команды  1;
end
else
begin
  Команды  2;
end;
```

Оператор ветвления может быть в полной или в сокращенной форме. В сокращенной форме отсутствует блок else:

```
if <условие> then
begin
  Команды;
end;
```

Условие в записи оператора ветвления бывает простым и составным. Операторные скобки могут быть опущены, если внутри их находится одна команда.

Пример 16.1. Задано число x. Нужно определить, является ли оно положительным или нет, и вывести соответствующее сообщение.

Этапы выполнения задания

I. Исходные данные: x (введенное число).

II. Результат: соответствующее сообщение.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Проверка значения выражения ($x > 0$).

3. Вывод результата.

IV. Описание переменных: x — integer.

16.2. Решение задач с использованием оператора ветвления

Пример 16.2. В момент времени 00:00 на светофоре для пешеходов включили зеленый сигнал. Далее сигнал светофора сменяется каждую минуту: 1 минуту горит зеленый сигнал, 1 минуту — красный. Известно, что с момента включения светофора прошло

Пример 16.1. *Продолжение.*

VI. Тестирование.

Запустить программу и ввести значение $x = 5$. Результат:

Окно вывода

```
Введите x = 5
положительное
```

Запустить программу и ввести значение $x = -1$. Результат:

Окно вывода

```
Введите время x = -1
не положительное
```

VII. Анализ результатов. Для полной проверки программы требуется еще проверить значение $x = 0$.

Окно вывода

```
Введите x = 0
не положительное
```

Пример 16.2.

V. Программа:

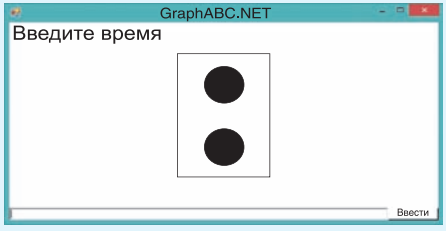
```
uses GraphABC;
var m:integer;
begin
  Rectangle(250,50,390,250);
  SetBrushColor(clBlack);
  Circle(320,100,30);
  Circle(320,200,30);
  SetBrushColor(clWhite);
  writeln('Введите время');
  read(m);
  writeln(m)[1];
  if m mod 2 = 1 then
    FloodFill(320,100,clRed)
  else
    FloodFill(320,200,clGreen);
end.
```

[1] При вводе данных в графическом окне они не отображаются на экране. Для того чтобы видеть, что ввели, необходимо дополнительно вывести введенное значение.

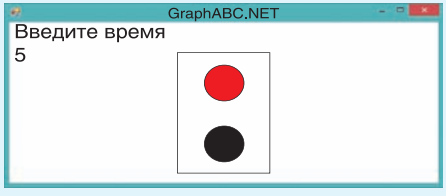
Пример 16.2. *Продолжение.*

VI. Тестирование.

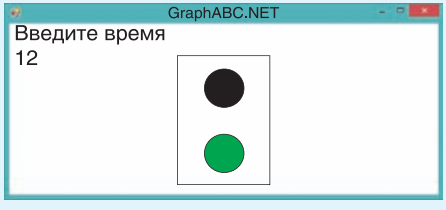
Вид графического окна до ввода числа:



Ввести значение $x = 5$. Результат:



Ввести значение $x = 12$. Результат:



Пример 16.3.

V. Программа:

```
Var x1, y1, x2, y2, r_T, r_K: real;
begin
  writeln('Танин дом'); read(x1,y1);
  writeln('Катин дом'); read(x2,y2);
  r_T := sqrt(x1*x1+y1*y1);
  r_K := sqrt(x2*x2+y2*y2);
  if r_T < r_K then
    writeln('Танин дом ближе')
  else
    writeln('Катин дом ближе');
end.
```

VI. Запустить программу и ввести значения: Танин дом — $x1 = 2.3$, $y1 = 4.5$, Катин дом — $x2 = -2.1$, $y2 = 4.9$

m минут. Требуется нарисовать светофор с включенным сигналом в соответствии с введенным значением времени.

Этапы выполнения задания

I. Исходные данные: m (заданное время).

II. Результат: рисунок светофора, зависящий от значения m.

III. Алгоритм решения задачи.

1. Рисование светофора (прямоугольник и 2 круга) с выключенными сигналами.

2. Ввод исходных данных.

3. Цвет сигнала будет зависеть от того, четным или нечетным будет значение m. Если m четное — сигнал зеленый (закрашиваем нижний круг), если нечетное — красный (закрашиваем верхний круг).

4. Закрасим нужный круг цветом в зависимости от четности m.

IV. Описание переменных: m — `integer`.

Пример 16.3. Таня и Катя живут в разных домах. Им стало интересно, кто из них живет ближе к школе. Они разместили на карте прямоугольную систему координат так, чтобы школа имела координаты (0; 0). Известно, что Танин дом имеет координаты ($x1$; $y1$), а Катин ($x2$; $y2$). Девочки ходят в школу по прямой и проходят разные расстояния. Нужно написать программу, которая определит, чей дом ближе к школе.

Этапы выполнения задания

I. Исходные данные: координаты домов девочек $x1$, $y1$, $x2$, $y2$.

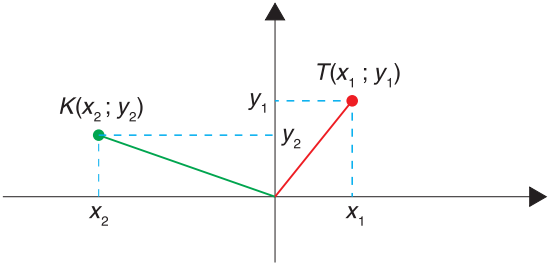
II. Результат: сообщение о том, чей дом ближе.

III. Алгоритм решения задачи.

1. Ввод координат домов.

2. Вычисление расстояний до школы: r_T (от Таниного дома) и r_K (от Катиного дома). Для вычисления воспользуемся теоремой Пифагора:

$$r\_T = \sqrt{x_1^2 + y_1^2} \quad \text{и} \quad r\_K = \sqrt{x_2^2 + y_2^2}.$$



3. Сравнение расстояний. Вывод ответа.

IV. Описание переменных: $x1$, $y1$, $x2$, $y2$, r_T, r_K имеют тип `real`.

Пример 16.4. Вася начал заниматься стрельбой из лука. Для тренировки он решил создать модель мишени, которая будет реагировать на лазер. Мишень представляет собой два круга (стреляет Вася пока не очень хорошо) разного радиуса с общим центром. Если Вася попал в маленький круг, то круг загорается зеленым. Большой круг при попадании в него загорается желтым. Если Вася не попал ни в один из кругов, то область вне кругов загорается красным. Необходимо реализовать компьютерную модель Васиной мишени (при попадании на границу круга ничего не должно происходить).

Этапы выполнения задания

I. Исходные данные: координаты точки выстрела (x; y).

II. Результат: рисунок мишени.

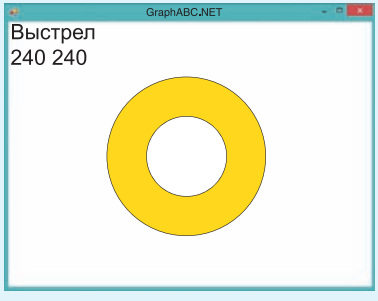
III. Алгоритм решения задачи.

Пример 16.3. *Продолжение.*
Результат должен быть таким:

Окно вывода
```
Танин дом
2.3 4.5
Катин дом
-2.1 4.9
Танин дом ближе
```

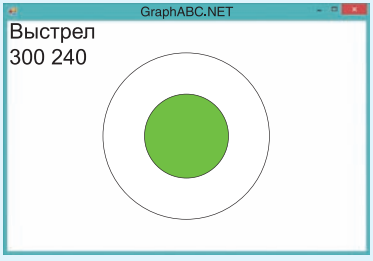
Пример 16.4.
V. Программа:
```
uses GraphABC;
var x,y, x0, y0, R_b, R_m, z:
integer;
begin
  x0 := 320; y0 := 240;
  R_b := 150; R_m := 75;
  Circle(x0,y0,R_b);
  Circle(x0,y0,R_m);
  writeln('Выстрел');
  read(x,y);
  writeln(x,' ',y);
  z := sqr(x-x0)+sqr(y-y0);
  if z < sqr(R_m) then
    FloodFill(x,y,clLightGreen)
  else
    if z < sqr(R_b) then
      FloodFill(x,y,clYellow)
    else
      FloodFill(x,y,clRed);
end.
```
VI. Тестирование.

Запустить программу и ввести координаты выстрела (240; 240). Результат:

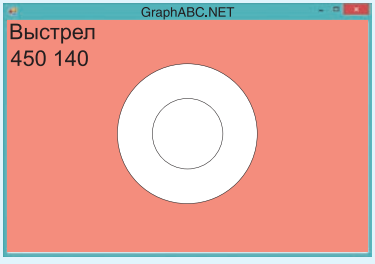


Пример 16.4. *Продолжение.*

Запустить программу еще раз и ввести координаты выстрела (300; 240).



Запустить программу еще раз и ввести координаты выстрела (450; 140).



Пример 16.5.

V. Программа:

```
var a, a1, a2, a3: integer;
begin
  write('Введите a = ');
  read(a);
  if (a > 99) and (a < 1000) then
  begin
    //Первая цифра
    a1 := a div 100;
    //Вторая цифра
    a2 := a mod 100 div 10;
    //Третья цифра
    a3 := a mod 10;
    writeln(a1);
    writeln(a2);
    writeln(a3);
  end
  else
    writeln('не трехзначное');
end.
```

1. Рисование мишени: 2 круга радиусов $R\_b = 150$ и $R\_m = 75$ с центром в точке $(x_0; y_0)$, $x_0 = 320$, $y_0 = 240$. Сначала рисуем круг большего радиуса.

2. Ввод данных: координаты точки выстрела.

3. Цвет рисунка будет зависеть от того, в какую область относительно кругов попала точка. Возможны 3 случая:

1) точка внутри маленького круга. Длина отрезка между точкой и центром круга меньше радиуса. По теореме Пифагора:

$(x - x_0)^2 + (y - y_0)^2 < R\_m^2$;

2) если условие а) не выполняется, проверяем, принадлежит ли точка большому кругу:

$(x - x_0)^2 + (y - y_0)^2 < R\_b^2$;

3) если условия а) и б) не выполняются, то Вася не попал в мишень.

4. Для сокращения записи определим переменную $z = (x - x_0)^2 + (y - y_0)^2$.

5. Закрасим нужную область цветом в зависимости от проверки условий.

IV. Описание переменных: x, y, x_0, y_0, R_b, R_m, z имеют тип `integer`.

Пример 16.5. Проверить, является ли введенное число трехзначным, и если да, то вывести цифры этого числа в отдельных строках.

Этапы выполнения задания

I. Исходные данные: a (трехзначное число).

II. Результат: переменные $a1$, $a2$, $a3$ (цифры числа) или сообщение «не трехзначное».

III. Алгоритм решения задачи.

1. Ввод исходного числа.

2. Проверка числа. Число a является трехзначным, если $99 < a < 1000$.

3. Если число трехзначное, выделяем его цифры:

1) для выделения первой цифры $a1$ находим целую часть от деления числа a на 100;

2) для выделения второй цифры $a2$ числа a находим остаток от его деления на 100, а затем целую часть от деления полученного остатка на 10;

3) последняя цифра числа $a3$ является остатком от деления числа a на 10.

4. Вывод результата.

IV. Описание переменных: все переменные имеют тип `integer`.

Пример 16.5. *Продолжение.*

VI. Тестирование.

Запустить программу и ввести значение 345.

Результат следующий:

Окно вывода

```
Введите a = 345
3
4
5
```

Другой вариант исходных данных:

Окно вывода

```
Введите a = 24
не трехзначное
```

1. Что такое оператор ветвления?
2. Чем отличается полная запись оператора ветвления от сокращенной?
3. Можно ли использовать составные условия в операторе ветвления?

Упражнения

1 Можно ли изменить логическое выражение в операторе ветвления в примере 16.1 так, чтобы сообщения `'положительное'` и `'не положительное'` пришлось поменять местами? Если да, то как это сделать?

2* Какие изменения нужно внести в программу примера 16.1, чтобы для числа рассматривались три случая: `'положительное'`, `'отрицательное'`, `'равно нулю'`?

3 Подключите графический режим в программе примера 16.1. Измените программу так, чтобы сообщение `'положительное'` выводилось красным цветом, а сообщение `'не положительное'` — синим.

4* Измените программу в примере 16.2 так, чтобы четность (нечетность) числа проверялась с использованием функции odd.

5 Напишите программу. Задано число x. Если число четное, то нарисовать на экране зеленый прямоугольник, а если нечетное, то красный круг (см. пример 16.2).

6 Добавьте в программу из примера 16.3 проверку корректности исходных данных: координаты домов должны быть такими, чтобы расстояния до школы были разными. Если расстояния одинаковы, то вывести сообщение `'Координаты введены неверно'`, а если разные, то вывести ответ.

7 Какие изменения понадобится внести в программу из примера 16.3, если допустить, что девочки могут проходить одинаковые расстояния? Внесите изменения в программу и проверьте правильность ее работы.

8 Для усложнения тренировок Вася (пример 16.4) решил менять местоположение мишени и радиусы кругов. Добавьте в программу возможность ввода радиусов большого и маленького кругов, а также центра мишени. Проверьте правильность работы программы на различных наборах исходных данных.

9 Как известно, многие задачи имеют не единственное решение. Так, Юля нашла другой способ вычисления второй цифры трехзначного числа для примера 16.5. Какую из команд использовала Юля? Объясните, что получится при выполнении каждой из приведенных команд.

1) `a2:=a mod 10 div 10;` 2) `a2:=a div 10 mod 10;` 3) `a2:=a div 100 mod 10.`

10 Программу из примера 16.5 изменили. Сформулируйте условие задачи, которая решается с помощью этой программы.

```
var a, a1, a2, a3: integer;
begin
 write('Введите  a = '); read(a);
 if (a > 99) and (a < 1000) then
 begin
  // Первая цифра
  a1 := a div 100;
  // Вторая цифра
  a2 := a mod 100 div 10;
  // Третья цифра
  a3 := a mod 10;
  if a1 mod 2 = 0 then
     writeln(a1,'– четная ');
  if a2 mod 2 = 0 then
     writeln(a2,'– четная ');
  if a3 mod 2 = 0 then
     writeln(a3, '– четная');
  if odd(a1) and odd(a2) and odd(a3) then
     writeln('нет четных цифр');
 end
 else
  writeln('не трехзначное');
end.
```

11 Программу из задания 10 проверили для некоторых случаев. Все ли возможные ситуации рассмотрели? Что нужно добавить?

Окно вывода	Окно вывода	Окно вывода	Окно вывода
Введите a = 246 2 - четная 4 - четная 6 - четная	Введите a = 103 0 - четная	Введите a = 537 нет четных цифр	Введите a = 26 не трехзначное

12 Петя решил усовершенствовать программу из задания 10 и проверку цифр в числе записал следующим образом:

```
if a1 mod 2 = 0 then
 writeln(a1,' — четная')
else
 if a2 mod 2 = 0 then
   writeln(a2,' — четная')
 else
   if a3 mod 2 = 0 then
     writeln(a3, ' — четная')
   else
     writeln('нет четных цифр');
```

Почему Петина отметка оказалась невысокой? Приведите примеры, для которых программа выдает неправильный ответ. Приведите примеры, когда программа выдает правильный ответ, если такое возможно.

13 Дано натуральное число. Напишите программу, которая проверяет, является ли оно трехзначным и кратна ли 7 сумма его цифр.

14* Дано натуральное число. Напишите программу, которая проверяет, является ли оно четырехзначным и расположены ли его цифры в порядке убывания.

15* Вася научился попадать в центр мишени из примера 16.4 и решил перейти к более сложным тренировкам. Теперь его мишень представляет собой три вложенных круга с радиусами $R1$, $R2$, $R3$ (известно, что $R1 < R2 < R3$). Реализуйте компьютерную модель этой мишени. Цвета выберите самостоятельно.

§ 17. Оператор цикла

17.1. Оператор цикла с предусловием

Алгоритмическая конструкция *повторение (цикл)* представляет собой последовательность действий, выполняемых многократно (см. блок-схему в примере 13.2, с. 60). Саму последовательность называют **телом цикла.**

Оператор цикла — команда, реализующая алгоритмическую конструкцию *повторение* на языке программирования.

В Pascal существуют разные возможности управлять тем, сколько раз будет повторяться тело цикла. Может быть задано условие продолжения или

Цикл с заданным условием окончания работы в PascalABC записывается следующим образом:

```
repeat
  тело цикла;
until <условие>;
```

Цикл работает, пока условие ложно, и прекращает работу, когда условие становится истинным.

Этот цикл называют **циклом с постусловием**, так как проверка условия осуществляется после выполнения тела цикла. Цикл с постусловием всегда выполняется хотя бы один раз.

Циклы **repeat** и **while** в PascalABC взаимозаменяемы, поэтому при написании программ достаточно использования только одного из них.

окончания работы цикла, а также число повторений тела цикла.

Цикл с предусловием используется в том случае, когда известно условие продолжения работы. Для записи оператора цикла с предусловием используется команда **while**. Формат команды:

```
while <условие> do
begin
  тело цикла;
end;
```

Пример 17.1. Написать программу для рисования ряда окружностей с радиусом 10 пикселей вдоль верхнего края графического окна.

Этапы выполнения задания

I—II. Результатом работы программы, не зависящей от исходных данных, будет рисунок, отображающий ряд окружностей вдоль верхнего края графического окна.

III. Алгоритм решения задачи.

1. *Положение первой окружности*. Окружность расположим в верхнем левом углу. Для этого задается радиус $r = 10$ и координаты центра $x = 10$, $y = 10$.

2. *Положение любой другой окружности*, удовлетворяющей условию задачи, будет зависеть от координаты x. В цикле будем изменять значение x. Каждое новое значение будет на 20 (на размер диаметра) больше предыдущего.

3. Цикл должен завершиться, когда значение координаты x станет больше чем 640 — горизонтальный размер окна.

IV. Описание переменных: x, y, r — integer.

Пример 17.1.

V. Программа:

```
uses GraphABC;
var x, y, r: integer;
begin
  r := 10;
  x := 10; y := 10;
  while x < 640 do
  begin
    Circle(x, y, r);
    x := x + 20;
  end;
end.
```

VI. Тестирование

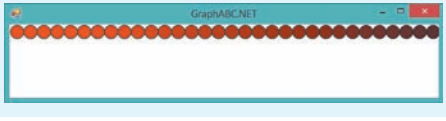
Запустить программу. Результат:

VII. Числовое значение в условии цикла можно заменить функцией, определяющей горизонтальное разрешение окна: WindowWidth:

```
while x < WindowWidth do
```

Функции RedColor, GreenColor, BlueColor позволяют менять интенсивность соответствующего цвета.

```
uses GraphABC;
var x, y, r, c: integer;
begin
  r := 10;
  x := 10; y := 10;
  c := 255;
  while x < 640 do
  begin
   //Интенсивность красного
    SetBrushColor(RedColor(c));
    Circle(x,y,r);
    x := x + 20;
   //Уменьшение интенсивности
    c := c-5;
  end;
end.
```



17.2. Оператор цикла с параметром

Цикл с параметром используется тогда, когда известно количество повторений.

Для записи оператора цикла с параметром используется команда **for**. Формат команды:

```
for var[1] i:= N1 to N2 do
begin
  тело цикла;
end;
```

Или

```
for var i:= N2 downto N1 do
begin
  тело цикла;
end;
```

В первом варианте параметр цикла i изменяется от *N1* до *N2*, каждый раз увеличиваясь на 1. Во втором — параметр i уменьшается на 1 при каждом выполнении тела цикла от *N2* до *N1*. Если *N1* > *N2*, цикл не выполняется ни разу. Изменять значение параметра внутри тела цикла нельзя.

Пример 17.2. Написать программу для вывода таблицы умножения на заданное число x.

Этапы выполнения задания

I. Исходные данные: x (введенное число).

II. Результат: 9 строк вида $a * x = c$.

III. Алгоритм решения задачи.

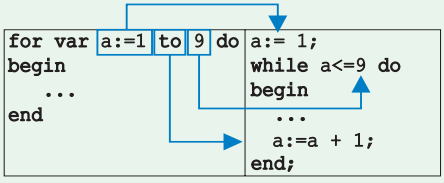
1. Значение переменной a изменяется в цикле от 1 до 9.

2. Значение переменной $c = a \cdot x$.

3. Так как количество повторений заранее известно, используем цикл **for**.

IV. Описание переменных: x, c — integer.

Любой цикл **for** может быть заменен на цикл **while**:



Обратное не всегда возможно.

Пример 17.2.

V. Программа:

```
var x, c : integer;
begin
 write('Введите x = '); read(x);
 for var a : = 1 to 9 do
 begin
  c := a * x;
  writeln(a,' * ',x,' = ',c);
 end;
end.
```

VI. Тестирование.

Запустить программу. Ввести $x = 7$.

Окно вывода

```
Введите x = 7
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
```

Решение с помощью цикла while:

```
var a, x, c : integer;
begin
 write('Введите x = '); read(x);
 a := 1;
 while a < = 9 do
 begin
  c := a * x;
  writeln(a,' * ',x,'= ',c);
  a := a + 1;
 end;
end.
```

VII. Проверить результат.

[1] Ключевое слово var может быть опущено, тогда переменная *i* должна быть описана (как integer) в разделе описания var до начала программы.

При решении задач с использованием оператора цикла важно правильно выбрать вид цикла. Если известно количество повторений тела цикла, то выбирают цикл **for**, а иначе — цикл **while**.

Внутри цикла можно использовать операторы `break` (немедленный выход из текущего цикла) и оператор `continue` (переход к концу тела цикла).

Пример 17.3.

V. Программа:

```
uses GraphABC;
var a,x1,y1,x2,y2: integer;
begin
  write('Введи a = ');
  read(a); write(a);
  x1 := 50; y1 := 50;
  x2 := 450; y2 := 450;
  for var i := 1 to 20 do
  begin
    Rectangle(x1,y1, x2,y2);
    x1 := x1 + a;  y1 := y1 + a;
    x2 := x2 - a;  y2 := y2 - a;
  end;
end.
```

VI. Тестирование.

Запустить программу и ввести значение $a = 10$. Результат:

17.3. Решение задач с использованием оператора цикла

Пример 17.3. Нарисовать 20 квадратов с общим центром. Длина стороны самого большого квадрата 400, верхний левый угол расположен в точке (50; 50). Координаты верхнего левого и нижнего правого углов каждого следующего квадрата изменяются на a (a — вводится).

Этапы выполнения задания

I. Исходные данные: a (введенное число).

II. Результат: рисунок, отображающий квадраты.

III. Алгоритм решения задачи.

1. Первым рисуется самый большой квадрат. Координаты его верхнего левого угла $x1 = 50$, $y1 = 50$. Координаты нижнего правого угла $x2 = 450$, $y2 = 450$.

2. Для определения положения другого квадрата нужно координаты верхнего левого угла увеличить на a, а нижнего правого — уменьшить на a.

3. Будем использовать цикл **for**, поскольку задано количество квадратов.

IV. Описание переменных: a, $x1$, $y1$, $x2$, $y2$ — `integer`.

Пример 17.4*. Вывести на экран наибольшее натуральное число из промежутка $[n, m]$, которое делится на заданное число x.

Этапы выполнения задания

I. Исходные данные: n, m (границы промежутка), x (заданное число).

II. Результат: искомое число или сообщение «Нет таких чисел».

III. Алгоритм решения задачи.

1. Пусть i — текущее число из промежутка.

2. Поскольку нас интересует наибольшее число из промежутка, то просмотр чисел начнем со значения $i = m$. На каждом шаге будем уменьшать i на 1.

3. Цикл завершится, если мы нашли число, делящееся на x без остатка (остаток равен нулю), или просмотрели все числа из промежутка $[n, m]$.

4. Так как количество повторений заранее неизвестно, используем цикл **while**.

Цикл будет продолжать работу до тех пор, пока условие, сформулированное в пункте 3, будет ложным. А именно: ложным должно быть условие `(i < n) or (i mod x = 0)`. Тогда условие not `((i < n) or (i mod x = 0))` будет истинным. Согласно правилам построения отрицаний (см. пример 15.5) это условие можно заменить условием: `(i >= n) and (i mod x <> 0)`. Его и будем использовать в качестве условия цикла.

5. Если по окончании цикла $i = n - 1$, то нет чисел, удовлетворяющих условию задачи.

IV. Описание переменных: n, m, x, i — `integer`.

Пример 17.4*.

V. Программа:

```
var i, n, m, x : integer;
begin
 writeln('Введите границы n, m');
 read(n,m);
 write('Введи  x = ');
 read(x);
 i := m;
 while (i >= n) and
       (i mod x <> 0) do
  i := i - 1;
 if i = n - 1 then
  writeln('Нет таких чисел')
 else
  writeln('Искомое число – ',i);
end.
```

VI. Тестирование.

Запустить программу и ввести значения $n = 10$, $m = 20$, $x = 3$. Результат:

Окно вывода

```
Введите границы n, m
10 20
Введи x = 3
Искомое число - 18
```

Запустить программу и ввести значения $n = 38$, $m = 45$, $x = 37$. Результат:

Окно вывода

```
Введите границы n, m
38 45
Введи x = 37
Нет таких чисел
```

1. Что такое оператор цикла?
2. Каким образом можно управлять количеством выполнений тела цикла?
3. Как записывается оператор цикла с предусловием?
4. Как записывается оператор цикла с параметром?

Упражнения

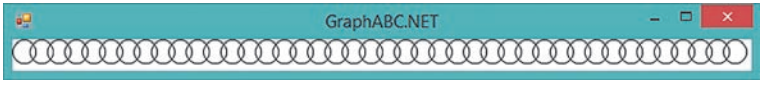
1 Измените программу из примера 17.1.

1. Радиусы окружностей равны 20.
2. Окружности располагаются вдоль левого края окна.
3. Радиус окружности вводится пользователем.

4. Окружности образуют рамку вокруг окна.

5*. Пользователь задает границу окна, вдоль которой будут располагаться окружности (например: 1 — верхняя, 2 — левая, 3 — правая, 4 — нижняя).

2 Какие изменения нужно внести в программу из примера 17.1 для того, чтобы рисунок выглядел следующим образом?



3 Внесите изменения в программу из примера 17.2. Пользователь задает значение второго множителя, а также начальное и конечное значения первого множителя.

4 При каком максимальном значении *a* на экране будут видны все 20 квадратов из примера 17.3? Почему при больших значениях *a* не видны все квадраты? Измените программу так, чтобы квадраты рисовались от самого маленького к самому большому (установите прозрачную заливку).

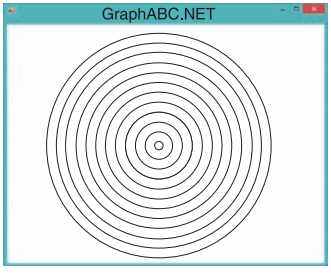
5 Какие изменения нужно внести в программу из примера 17.3, чтобы получить следующее изображение? Функции для изменения интенсивности цвета см. в примере 17.1.

6 Измените программу из примера 17.3. Длина стороны самого большого квадрата 400, а длина стороны каждого следующего квадрата на *x* меньше (*x* вводится).

7 Напишите программу, которая рисует ряд окружностей заданного радиуса, расположенных по диагонали графического окна. Рассмотрите два варианта:

1. Графическое окно квадратное.

2*. Графическое окно прямоугольное.



8 Напишите программу, которая рисует концентрические окружности с центром в середине графического окна. Радиус самой маленькой окружности — 10 пикселей. Разница радиусов — 20 пикселей. Используйте изменение интенсивности какого-либо цвета (или двух одновременно) для заливки кругов.

9 В магазине продают конфеты в упаковках по 0.1 кг, 0.2 кг, ... 0.9 кг, 1 кг. Известно, что 1 кг конфет стоит *x* рублей. Выведите стоимости каждой упаковки в виде:

0.1 кг конфет стоит ... р.;

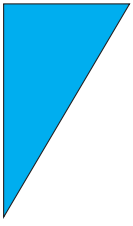
0.2 кг конфет стоит ... р. ... .

10* Выведите на экран наименьшее натуральное число из промежутка [*n*, *m*], которое является нечетным и не делится на введенное значение *x*.

§ 18. Составление алгоритмов для работы с графикой

18.1. Расчеты в графических построениях

Пример 18.1. Нарисовать прямоугольный треугольник, соответствующий рисунку (катеты треугольника параллельны осям координат). Длины катетов и координаты прямого угла вводятся.



Этапы выполнения задания

I. Исходные данные: a и b (длины катетов), x и y (координаты вершины прямого угла).

II. Результат: изображение прямоугольного треугольника.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Чтобы изобразить треугольник, нужно выполнить следующие действия:

1) построить линии из точки с координатой $(x; y)$ в точки с координатами $(x + a; y)$ и $(x; y + b)$;

2) соединить линией точки $(x + a; y)$ и $(x; y + b)$;

3) закрасить треугольник. Для закрашивания треугольника нужно знать координаты какой-либо точки внутри треугольника. Такой точкой в

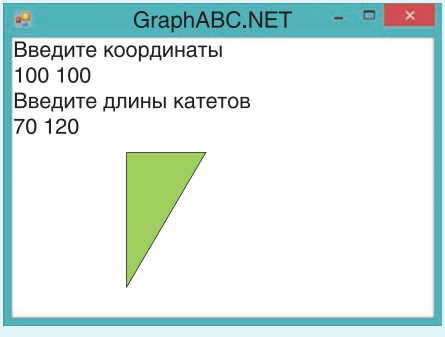
Пример 18.1.

V. Программа:

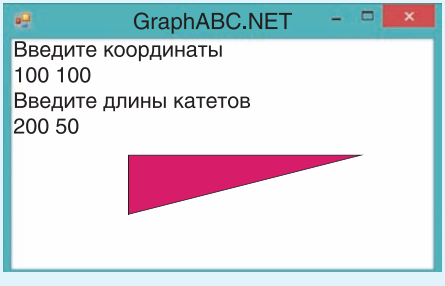
```
uses GraphABC;
var a,b,x,y,x_c, y_c:integer;
begin
 writeln('Введите координаты');
 read(x,y); writeln (x,' ',y);
 writeln('Введите длины катетов');
 read(a,b); writeln (a,' ',b);
 Line(x,y,x+a,y); Line(x,y,x,y+b);
 Line(x+a,y,x,y+b);
 //Координаты точки
 //Внутри треугольника
 x_c := x + 2; y_c := y + 2;
 FloodFill(x_c,y_c,clRandom);
end.
```

VI. Тестирование.

Запустить программу и ввести значения: координаты (100; 100), длины катетов 70 и 120. Результат:



Другой вариант:



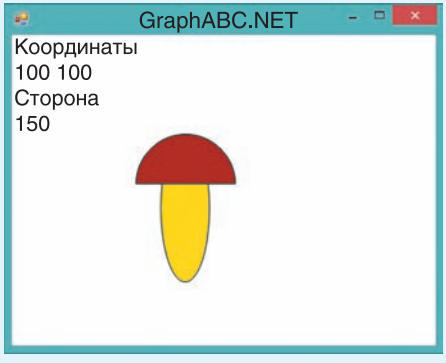
Пример 18.2*.

V. Программа:

```
uses GraphABC;
var x,y,d: integer;
begin
  writeln('Координаты');
  read(x,y);
  writeln (x,' ',y);
  writeln('Сторона');
  read(d);
  writeln (d);
  SetBrushColor(clYellow);
  Ellipse (x + d div 3,y,
           x + 2 * d div 3,y + d);
  SetBrushColor(clBrown);
  Pie(x + d div 2,y + d div 3,
      d div 3,0,180);
end.
```

VI. Тестирование.

Запустить программу и ввести значения: координаты (100; 100), сторона 150. Результат:



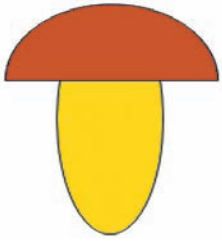
* Гриб можно вписывать не в квадрат, а в прямоугольник. В этом случае нужно задавать две величины, определяющие размеры прямоугольника: длину ($d1$) и ширину ($d2$).

данном случае может быть точка с координатами $(x + 2; y + 2)$[1].

IV. Описание переменных: все переменные имеют тип `integer`.

Многие графические построения можно обобщить, если предположить, что фигура должна быть вписана в квадрат. В этом случае для построения фигуры достаточно задать координаты $(x; y)$ верхнего левого угла квадрата и длину его стороны — d. Используя эти величины, можно получить координаты других вершин квадрата: $(x + d; y)$, $(x; y + d)$, $(x + d; y + d)$. Можно получить координаты середины стороны $(x + d \text{ div } 2; y)$ или центра квадрата $(x + d \text{ div } 2; y + d \text{ div } 2)$.

Пример 18.2*. Нарисовать в графическом окне гриб. Задать координаты верхнего левого угла квадрата и длину его стороны для определения местоположения и размеров гриба.



Этапы выполнения задания

I. Исходные данные: x и y — координаты верхнего левого угла квадрата, в который вписан гриб, d — длина его стороны.

II. Результат: изображение гриба.

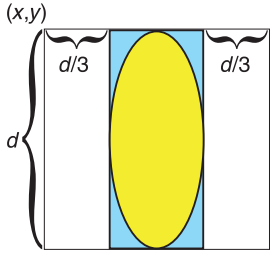
[1] Для произвольного треугольника можно воспользоваться формулами $\left(\frac{x_1 + x_2 + x_3}{3}; \frac{y_1 + y_2 + y_3}{3}\right)$.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Для того чтобы построить гриб, нужно выполнить следующие действия:

1) построить овал для изображения ножки гриба. Параметры команды для изображения эллипса определим следующим образом: `ellipse(x + d div 3, y, x + 2*d div 3, y + d);`



2) нарисовать шляпку гриба. Для этого можно использовать команду `Pie` (построение сектора круга). Координаты центра $\left(x + \frac{d}{2}; y + \frac{d}{3}\right)$, радиус — $\frac{d}{3}$. Углы равны 0° и 180° соответственно.

IV. Описание переменных: все переменные имеют тип `integer`.

Случайные числа имеют широкое применение в программировании. Они используются в шифровании и в моделировании. Многие компьютерные игры используют случайные числа. На основе случайных чисел генерируются капчи и пароли, реализуются различные лотереи.

В PascalABC для получения случайного числа используют функцию `random`. Способы записи функции:

`Random(a, b);` — возвращает случайное целое в диапазоне от a до b;

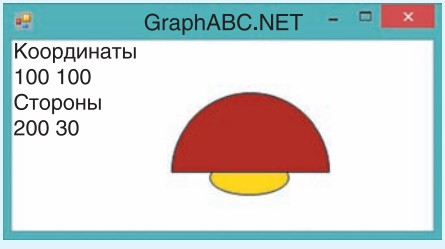
`Random(a);` — возвращает случайное целое в диапазоне от 0 до $a - 1$;

Пример 18.2*. *Продолжение.*

В программу рисования гриба нужно внести изменения, позволяющие рассчитать положение ножки и шляпки гриба относительно координат $(x; y)$ и величин $d1$ и $d2$.

```
var x,y,d1,d2: integer;
  ...
  writeln('Стороны');
  read(d1, d2);
  writeln (d1, ' ',d2);
  SetBrushColor(clYellow);
  Ellipse(x+d1 div 3, y,
    x + 2 * d1 div 3, y + d2);
  SetBrushColor(clBrown);
  Pie(x + d1 div 2, y + d2 div 3,
     max(d1,d2) div 3, 0, 180);
```

Результат:



Случайное число — число, которое принимает одно значение из множества, причем появление того или иного значения нельзя точно предсказать. Например, если бы числа появились в результате вытягивания бочонков в лото, то такая последовательность чисел была бы случайной.

В языках программирования используют **псевдослучайные числа**, которые получают с использованием генератора случайных чисел – алгоритма, порождающего последовательность чисел, элементы которой почти независимы друг от друга и обычно распределены равномерно на заданном интервале.

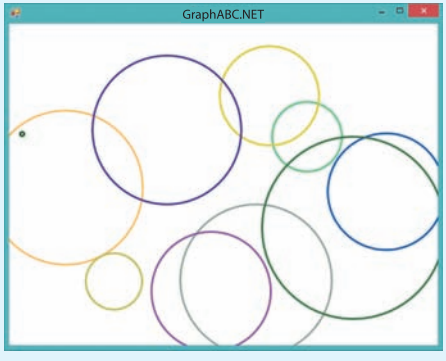
Пример 18.3.

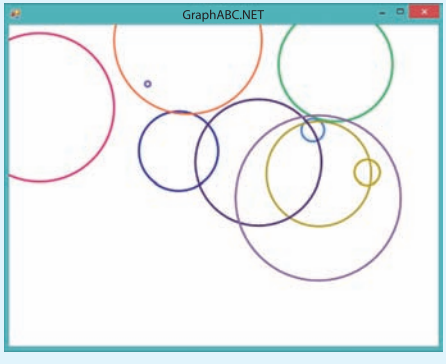
V. Программа:

```
uses GraphABC;
var x,y,r: integer;
begin
  SetPenWidth(3);
  SetBrushStyle(bsClear);
  for var i:= 1 to 10 do
  begin
    x := random(600);
    y := random(400);
    R := random(150);
    SetPenColor(clRandom);
    circle(x,y,r);
  end;
end.
```

VI. Тестирование.

Запустить программу. Должно быть нарисовано 10 окружностей. Разные варианты работы программы:





`Random;` — возвращает случайное вещественное в диапазоне [0..1).

Функция `clRandom` позволяет задать случайный цвет.

Пример 18.3. Написать программу для рисования на экране 10 разноцветных окружностей. Расположение окружностей, их радиусы и цвет определяются случайным образом.

Этапы выполнения задания

I—II. Результатом работы программы, не зависящей от исходных данных, будет изображение 10 окружностей.

III. Алгоритм решения задачи.

1. Установим толщину линий в 3 пикселя и прозрачную заливку.

2. Значения координат центра окружности и ее радиуса определяются функцией `random`. Значение цвета для границы круга — `clRandom`.

3. Так как количество повторений известно, будем использовать цикл **for**.

IV. Описание переменных: x, y (координаты центра), r (радиус) имеют тип `integer`.

18.2. Использование вспомогательных алгоритмов

Построение фигур можно оформлять в виде вспомогательных алгоритмов. Это позволит использовать такие алгоритмы для решения других задач.

Все графические процедуры, которые использовались ранее, имели параметры. Они позволяли определять местоположение и размер фигур. Пользователь также может составить свой вспомогательный алгоритм с параметрами.

Общий вид процедуры с параметрами:

```
procedure <имя> (<список
                параметров>:тип);
var ...
begin
 <команды>
end;
```

При вызове процедуры важно помнить, что количество параметров и их порядок должны соответствовать тому, как процедура описана.

Пример 18.4. Написать программу для построения n равнобедренных прямоугольных треугольников с длиной катета a. Расположение треугольников определяется случайным образом.

Этапы выполнения задания

I. Исходные данные: n (количество треугольников), a (длина катета).

II. Результат: изображение n треугольников.

III. Алгоритм решения задачи.

1. В примере 18.3 изображали окружности, расположенные случайным образом, а в примере 18.1 — прямоугольные треугольники. Воспользуемся программами этих примеров.

2. Ввод значений переменных n и a.

3. Так как количество повторений известно, будем использовать цикл **for**.

4. Изменим программу из примера 18.3. Для этого команду `cirle` (построение окружности) заменим на команду построения треугольника:

1) местоположение треугольника задается координатами прямого угла, которые определим случайным образом;

2) катеты прямоугольного треугольника имеют одинаковую длину — значение a.

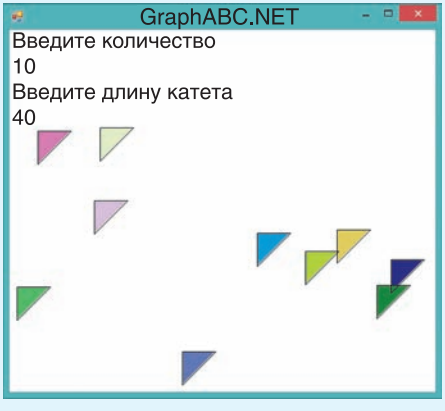
Пример 18.4.

V. Программа:

```
uses GraphABC;
var n, x, y, a : integer;
procedure pr_treug (x, y, a,
          b : integer);
var x_c, y_c:integer;
begin
 line(x, y, x + a,y);
 line(x, y, x, y + b);
 line(x + a, y, x, y + b);
 x_c := x + 2; y_c := y + 2;
 FloodFill(x_c,y_c,clRandom);
end;
begin
 writeln('Введите количество');
 read(n); writeln (n);
 writeln('Введите длину катета');
 read(a); writeln (a);
 for var i:= 1 to n do
 begin
  x:= random(500);
  y:= random(400);
  pr_treug(x, y, a, a);
 end;
end.
```

VI. Тестирование.

Запустить программу. Результат:

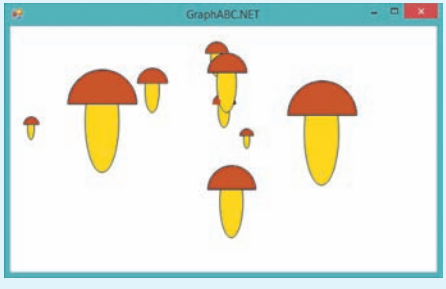


При вводе других значений результаты будут иными.

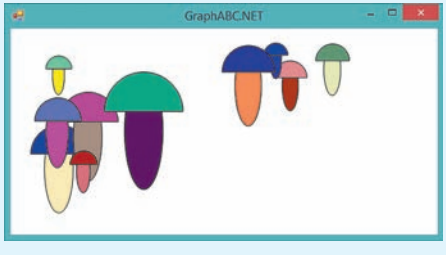
Пример 18.5*.
V. Программа:

```
uses GraphABC;
var x,y,d: integer;

procedure grib(x,y,d:integer);
begin
  SetBrushColor(clYellow);
  Ellipse(x + d div 3, y,
       x + 2*d div 3, y + d);
  SetBrushColor(clBrown);
  Pie(x + d div 2, y + d div 3,
        d div 3, 0, 180);
end;

Begin
  for var i: = 1 to 10 do
  begin
    x:= random(400);
    y:= random(200);
    d:= random(150);
    grib(x,y,d);
  end;
end.
```

VI. Тестирование.
Результат может быть следующим:



Можно добавить раскраску случайным цветом:



5. Построение одного прямоугольного треугольника опишем во вспомогательном алгоритме `pr_treug`. Параметры процедуры построения треугольника — координаты вершины прямого угла и длины катетов. Алгоритм описан в примере 18.2.

IV. Описание переменных: все переменные имеют тип `integer`.

Пример 18.5*. Нарисовать 10 грибов. Расположение и их размеры определяются случайным образом.

Этапы выполнения задания

I—II. Результатом работы программы, не зависящим от исходных данных, будет изображение 10 грибов.

III. Алгоритм решения задачи.

1. Построение одного гриба опишем во вспомогательном алгоритме. Алгоритм описан в примере 18.2.

2. Значения координат верхнего левого угла и размер гриба определяются функцией `random`.

3. Так как количество повторений известно, будем использовать цикл **for**.

IV. Описание переменных: x, y (координаты верхнего левого угла), d (размер) — `integer`.

Пример 18.6. Заполнить графическое окно окружностями с радиусом 10.

Этапы выполнения задания

I—II. Исходные данные отсутствуют. Окружности должны заполнить все графическое окно.

III. Алгоритм решения задачи.

1. Задача является обобщением задачи примера 17.1. Команды программы следует повторить для нескольких рядов окружностей. Количество рядов определяется высотой окна. Рисование

одного ряда оформим как вспомогательный алгоритм `row`.

2. Положение любого ряда окружностей определяется координатой y. Для каждого значения y, пока он не станет большим, чем вертикальный размер экрана, выполняем в цикле следующее:

1) рисуем ряд окружностей;

2) изменяем y.

IV. Описание переменных: x, y, r — `integer`.

В примере 18.6 показано, как заполнить графическое окно окружностями. Внеся небольшие изменения в эту программу, можно заполнять графическое окно любыми другими фигурками. Для этого достаточно заменить команду `Circle(x,y,r)` в процедуре `row` на другую команду. Можно выбрать графический примитив из библиотеки GraphABC или самостоятельно написать процедуру рисования фигурки (например, использовать процедуру рисования гриба из примера 18.5).

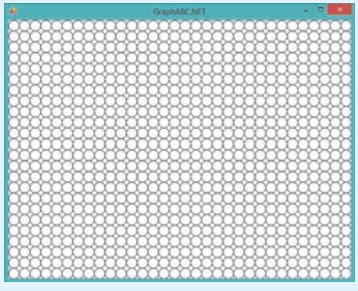
Пример 18.6.

V. Программа:

```
uses GraphABC;
var x, y, r : integer;
procedure row(y : integer);
begin
  x :=10; R := 10;
  while x < WindowWidth do
  begin
    Circle(x,y,r); x := x+20;
  end;
end;
begin
  y := 10;
  while y < = WindowHeight do
  begin
    Row(y); y := y + 20;
  end;
end.
```

VI. Тестирование.

Результат:



1. Как задать случайное число?
2. Как задать случайный цвет?
3. Как описать процедуру с параметрами?

Упражнения

1 Выполните задания для примера 18.1.

1. Поэкспериментируйте с программой, вводя различные значения исходных данных.

2. Объясните, что происходит при вводе отрицательных значений длин катетов.

3. Что произойдет, если ввести отрицательные значения координат? Объясните результат.

2 Выполните задания для примера 18.3.

1. Выполните программу несколько раз. Уберите прозрачную заливку. Объясните, почему некоторые окружности не видны.

2. Внесите в программу такие изменения, чтобы можно было изобразить 20 кругов; 100 кругов.

3. Какой максимальный размер может иметь радиус круга в программе? Внесите в программу изменения так, чтобы рисовались круги с радиусом не более 20. Количество кругов установите равным 10 000.

4. Внесите изменения в программу так, чтобы пользователь мог вводить количество отображаемых на экране кругов.

3 Выполните задания для примера 18.4.

1. Запустите программу несколько раз. Объясните, почему при некоторых запусках треугольники рисуются поверх текста в верхнем левом углу экрана. Измените программу так, чтобы треугольники рисовались ниже текста (правее текста).

2. Добавьте в программу возможность ввода длины второго катета.

3. Измените программу так, чтобы длины катетов задавались случайным образом.

4 Напишите программу, которая строит случайным образом изображения 20 горизонтальных отрезков длиной 30 пикселей. Разработайте два варианта решения задачи. Один с использованием цикла **`while`**, а другой — цикла **`for`**.

1. Сравните две программы решения задачи. Какой вариант решения данной задачи представляется вам лучшим? Почему?

2. Задайте в программе толщину отрезка в 3 пикселя.

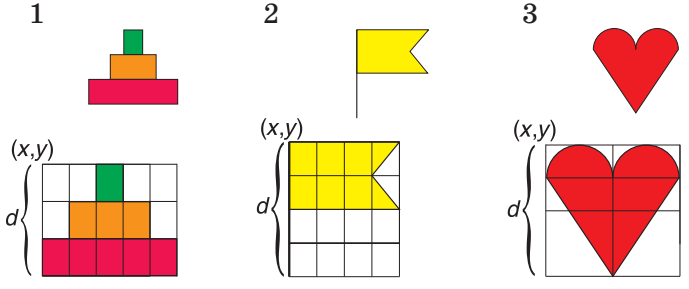
3. Какие изменения нужны в программе, чтобы толщина отрезка была случайным числом из промежутка [2; 8]?

4. Внесите изменения в программу так, чтобы пользователь мог вводить количество отображаемых на экране отрезков.

5. Какие изменения нужно внести в программу, чтобы вместо горизонтальных отрезков изображались вертикальные? Диагональные?

5 Используя процедуру рисования треугольника из примера 18.4, нарисуйте ряд треугольников вдоль верхнего (левого) края графического окна.

6* Напишите программу для рисования одной из фигурок. Задаются координаты верхнего левого угла и длина стороны квадрата (длины сторон прямоугольника):



7* Напишите программу, которая нарисует ряд фигурок вдоль края графического окна. Используйте фигурки, которые рисовали в задании 6.

8 Выполните задания для примера 18.6.

1. Измените в программе значение $r = 10$ на $r = 20$. Почему получился такой рисунок? Поэкспериментируйте со значениями радиуса, установив прозрачную заливку.
2. Какие изменения нужно внести в программу, чтобы экран заполнялся кругами с радиусом 20 без пересечений?
3. Внесите изменения в программу так, чтобы все круги были красными или разноцветными.
4. Внесите в программу изменения так, чтобы графическое окно можно было заполнять кругами введенного радиуса.

9* Напишите программу, которая заполнит все графическое окно:

1. Грибами (пример 18.2).
2. Фигурками из задания 6.

§ 19. Использование основных алгоритмических конструкций для решения практических задач

19.1. Использование числовых последовательностей

Числовые последовательности позволяют описывать многие процессы, происходящие в природе и обществе.

Например, последовательность чисел 2, –1, 0, 2, 0, 1, –2 может задавать значения температуры по дням недели; последовательность 746, 751, 758 — значения средней заработной платы сотрудников предприятия за квартал и т. д.

Последовательности могут задаваться формулой, в которой значение элемента зависит от того, какой у него номер в последовательности (пример 19.1).

Другим способом задания элементов последовательности является определение значения нового элемента

Подтверждением важности числовых последовательностей является тот факт, что создана целая энциклопедия числовых последовательностей[1].

Пример 19.1. Элементы последовательности нечетных положительных чисел можно описать с помощью формулы $a_n = 2n - 1$. В этой формуле n — номер элемента в последовательности. Минимальное значение числа $n = 1$. Используя формулу, получим последовательность: 1, 3, 5, 7, ...

Элементы последовательности могут быть действительными числами. Например, формула $a_n = \frac{n}{n^2 + 1}$ задает следующую последовательность: 0.5, 0.4, 0.3, 0.235, 0.192, ...

[1] Онлайн-энциклопедия целочисленных последовательностей. Режим доступа: http://oeis.org/?language=russian (дата доступа: 17.01.2018).

Пример 19.2. Одной из наиболее известных последовательностей является последовательность Фибоначчи: 1, 1, 2, 3, 5, 8, 13, ... Несложно заметить, что каждый ее элемент, начиная с третьего, равен сумме двух предыдущих. Это можно записать так: $f_n = f_{n-1} + f_{n-2}$, $f_1 = 1$, $f_2 = 1$.

Пример 19.3. Рассмотрим последовательность 2, 4, 8, 16, ... Каждое число в этой последовательности является степенью числа 2, поэтому можно задать последовательность формулой $a_n = 2^n$. С другой стороны, каждый элемент последовательности, начиная со второго, в два раза больше предыдущего. Получим формулу $a_n = 2a_{n-1}$ (для $n > 1$, $a_1 = 2$).

Пример 19.4.

V. Программа:

```
var k, a: integer;
begin
 write('Количество k = '); read(k);
  for var n := 1 to k do
  begin
    a:= 2*n; write(a, ' ');
  end;
end.
```

VI. Тестирование.

Запустить программу и ввести значения $k = 5$. Результат:

Окно вывода

```
Количество k = 5
2 4 6 8 10
```

Запустить программу и ввести значение $k = 100$. Результат:

Окно вывода

```
Количество k = 100
2 4 6 8 10 12 14 16 18 20 22 24 26 28
30 32 34 36 38 40 42 44 46 48 50 52
54 56 58 60 62 64 66 68 70 72 74 76
78 80 82 84 86 88 90 92 94 96 98 100
102 104 106 108 110 112 114 116 118
120 122 124 126 128 130 132 134 136
138 140 142 144 146 148 150 152 154
156 158 160 162 164 166 168 170 172
174 176 178 180 182 184 186 188 190
192 194 196 198 200
```

через значение предыдущего (пример 19.2).

Есть последовательности, которые можно задавать как первым, так и вторым способом (пример 19.3). Последовательности могут строиться из случайных чисел.

Пример 19.4. Вывести на экран первые k четных чисел.

Этапы выполнения задания

I. Исходные данные: k (количество чисел).

II. Результат: k четных числа, начиная с 2.

III. Алгоритм решения задачи.

1. Ввод числа k.

2. Для получения четного числа запишем формулу $a_n = 2n$.

3. Так как количество чисел заранее известно, то для их поучения можно воспользоваться циклом **for**.

4. Текущее число будем хранить в переменной a. Значение a вычисляется по формуле и зависит от значения n — счетчика цикла. Переменная n будет изменяться от 1 (номер первого четного числа) до k (номер последнего числа).

5. Полученные числа будем выводить в цикле через пробел.

IV. Описание переменных: k, a — `integer`.

Пример 19.5. Вывести на экран все элементы последовательности Фибоначчи меньше x (x вводится).

Этапы выполнения задания

I. Исходные данные: x (граница для чисел).

II. Результат: числа Фибоначчи меньше x.

III. Алгоритм решения задачи.

1. Ввод числа x.

2. Вывод первых двух элементов.

3. Числа Фибоначчи, начиная с третьего, получают по формуле $a_n = a_{n-1} + a_{n-2}$ ($a_1 = a_2 = 1$). Для вывода чисел понадобятся три переменные: значение a, которое нужно вывести, и два предыдущих значения — b и c.

n	1	2	3	4	5	6	7
Числа Фибоначчи	1	1	2	3	5	8	13
Текущий шаг			c	b	a		
Следующий шаг				c	b	a	

После вывода значения a нужно «сдвинуть» значения переменных: $c := b$; $b := a$. Начальные значения: $c := 1$; $b := 1$; $a := 2$.

4. Так как количество чисел заранее неизвестно, то для их вычисления нужно использовать цикл **while**. Условие продолжения работы цикла: $a < x$.

5. В цикле выводим текущее значение a, «сдвигаем» значения переменных и получаем новое значение a.

IV. Описание переменных: x, a, b, c — `integer`.

Числа Фибоначчи обладают множеством интересных математических свойств. Например:

1. Могут быть вычислены по формуле Бине: $f_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$.

2. Отношение следующего числа к предыдущему является постоянной величиной ≈ 1.618 (начиная с 13-го). Это число называют золотым сечением.

3. Для трех последовательных чисел Фибоначчи верно соотношение Кассини: $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$.

Пример 19.5.

V. Программа:

```
var a, b, c, x: integer;
begin
  write('Граница x = '); read(x);
  c := 1; b := 1; a := 2;
  write(c, ' ', b, ' ');
  while a < x do
  begin
    write(a, ' ');
    c := b; b := a; a := b + c;
  end;
end.
```

VI. Тестирование.

Запустить программу и ввести значение $x = 100$. Результат:

Окно вывода

```
Граница x = 100
1 1 2 3 5 8 13 21 34 55 89
```

Чтобы узнать, сколько чисел получили в качестве результата, определим переменную k. Переменная будет увличивать свое значение на 1 каждый раз, когда выводится очередное число. После завершения цикла можно вывести значение k.

```
var a, b, c, x, k: integer;
begin
  write('Граница x= ');
  read(x);
  c := 1; b := 1; a := 2;
  write(c, ' ', b, ' ');
  k := 2;
  while a < x do
  begin
    write(a, ' '); k := k+1;
    c := b; b := a; a := b + c;
  end;
  writeln;
  write('k = ',k);
end.
```

Результат для $x = 1000$.

Окно вывода

```
Граница x = 1000
1 1 2 3 5 8 13 21 34 55 89 144 233
377 610 987
k = 16
```

Пример 19.6.

V. Программа:

```
var n, k, a:integer;
begin
  write('Количество билетов n = ');
  read(n); k:= 0;
  for var i:= 1 to n do
  begin
    a:= random(1,100); write(a,' ');
    if a mod 5 = 0 then
      k := k+1;
  end;
  writeln;
  writeln('Выиграло ',k,
          ' билета(-ов)');
end.
```

VI. Тестирование. Запустить программу и ввести значение $n = 20$. Результат:

Окно вывода

```
Количество билетов n = 20
56 81 10 34 3 20 43 34 57 54 92 59 47
23 44 90 83 79 29 91
Выиграло 3 билета(-ов)
```

При одном и том же значении n программа может выдавать различные результаты, поскольку числа получаются случайным образом.

Пример 19.7.

V. Программа:

```
var m : integer;
 a, S: real;
begin
 write('Количество дней m = ');
 read(m); S := 0;
 for var n := 1 to m do
 begin
   a := n*n*n/(sqrt(n*n*n)-n+1);
   S := S+a;
 end;
 writeln('Всего бактерий =
         = ',S, ' млн');
end.
```

VI. Тестирование.

Запустить программу и ввести значение $m = 3$. Результат:

Окно вывода

```
Количество дней m = 3
Всего бактерий = 13.8230024772972 млн
```

Пример 19.6. Катя и Петя решили организовать благотворительную лотерею. Для этого они случайным образом генерируют номер билета. Номера билетов принадлежат промежутку [1..100]. Выигрышным билетом будет тот, номер которого кратен 5. Определить, сколько будет выигрышных билетов среди n сгенерированных Катей и Петей.

Этапы выполнения задания

I. Исходные данные: n (количество билетов).

II. Результат: k — количество выигрышных билетов.

III. Алгоритм решения задачи.

1. Ввод числа n.

2. До начала генерации количество выигрышных билетов равно нулю ($k := 0$).

3. Так как количество билетов заранее известно, то для получения их номеров можно воспользоваться циклом **for**.

4. Текущий номер билета будем хранить в переменной a. Номера билетов будем получать по формуле `a = Random(1,100)`и выводить на экран. Для каждого номера будем проверять, равен ли 0 остаток от деления числа на 5. И если равен, то увеличим на 1 значение переменной k.

5. Вывод результата.

IV. Описание переменных: n, k, a — `integer`.

19.2. Нахождение суммы элементов числовой последовательности

Пример 19.7. В лаборатории выводят полезные бактерии. Экспериментально

было установлено, что количество бактерий (в млн) зависит от номера дня, в который проводится эксперимент, следующим образом: $a_n = \frac{n^3}{\sqrt{n^3} - n + 1}$. Определите, сколько бактерий вывели за m дней.

Этапы выполнения задания

I. Исходные данные: m (число дней).

II. Результат: S (общее количество бактерий).

III. Алгоритм решения задачи.

1. Ввод числа m.

2. Для вычисления общего количества бактерий необходимо последовательно прибавлять количество бактерий, выведенных в текущий день, к уже полученному количеству бактерий. Начальное значение суммы равно 0.

3. Так как количество бактерий заранее известно, для вычисления суммы можно воспользоваться циклом **for**.

4. Количество бактерий в текущий день будем хранить в переменной a. Значение a зависит от значения n — счетчика дней. Переменная n изменяется от 1 до m.

5. Вывод результата S.

IV. Описание переменных: m — `integer`, S, a — `real`.

19.3. Возведение числа в степень

Пример 19.8. Возвести вещественное число a в целую степень n.

Этапы выполнения задания

I. Исходные данные: a (основание степени), n (показатель степени).

II. Результат: S (значение степени).

III. Алгоритм решения задачи.

Пример 19.7. *Продолжение.*

Для проверки правильности результата можно посчитать значение суммы на калькуляторе:

$$a_1 = 1; a_2 = \frac{2^3}{\sqrt{8} - 2 + 1} \approx 4.38;$$

$$a_3 = \frac{3^3}{\sqrt{27} - 3 + 1} \approx 8.44; S \approx 13.82.$$

Запустить программу и ввести значение $m = 30$. Результат:

Окно вывода

```
Количество дней m = 30
Всего бактерий = 2613.36198051392 млн
```

Пример 19.8.

V. Программа:

```
var n, m: integer;
    a, S: real;
begin
 write('Основание a = '); read(a);
 write('Показатель n = '); read(n);
 S := 1; m := abs(n);
 for var i:= 1 to m do
  S := S*a;
 if n<0 then S:= 1/S;
 writeln('Степень = ',S);
end.
```

VI. Тестирование программы.

Запустить программу и ввести значения $a = 5$, $n = 3$. Результат:

Окно вывода

```
Основание a = 5
Показатель n = 3
Степень = 125
```

Запустить программу и ввести значения $a = 3$, $n = 0$. Результат:

Окно вывода

```
Основание a = 3
Показатель n = 0
Степень = 1
```

Запустить программу и ввести значения $a = 5$, $n = -2$. Результат:

Окно вывода

```
Основание a = 5
Показатель n = -2
Степень = 0.04
```

VII. Правильность вычислений проверить на калькуляторе.

Пример 19.9.

V. Программа:

```
var k:integer;
   x,y,h:real;
begin
  writeln('Количество значений');
  readln(k); x := -3; h := 0.5;
  for var n:= 1 to k do
  begin
    y := (x+2)/(x*x+3);
    writeln(x:7:2,y:10:3); x := x+h;
  end;
end.
```

VI. Тестирование программы.

Запустить программу и ввести значение $k = 5$. Результат:

Окно вывода

```
Количество значений
5
  -3.00        -0.083
  -2.50        -0.054
  -2.00        -0.000
  -1.50        -0.059
  -1.00        -0.250
```

Добавим в программу вывод границ таблицы:

```
var k:integer;
   x,y,h: real;
begin
  writeln('Количество значений');
  readln(k); x := -3; h := 0.5;
  writeln('------------');
  writeln('|  x  |  y  |');
  writeln('------------');
  for var n := 1 to k do
  begin
   y := (x+2)/(x*x+3);
   writeln('|',x:7:2,'|',y:10:3,'|');
   x := x+h;
  end;
  writeln('------------');
end.
```

Результат при $k = 4$:

Окно вывода

```
Количество значений
4
---------------------------
|      x      |      y      |
---------------------------
|    -3.00    |    -0.083   |
|    -2.50    |    -0.054   |
|    -2.00    |     0.000   |
|    -1.50    |     0.095   |
---------------------------
```

1. Ввод чисел a, n.

2. Для возведения числа в целую неотрицательную степень нужно последовательно умножать на основание степени то значение, которое получили на предыдущем шаге. Это вытекает из равенства: $a^n = a^{n-1} \cdot a$.

3. Если степень отрицательная, то результат можно получить из равенства: $a^{-n} = \frac{1}{a^n}$. Количество повторений цикла m можно определить как $m = |n|$.

4. Для вычисления произведения можно воспользоваться циклом **for**. Начальное значение степени равно 1.

5. В конце проверим, является ли значение n положительным или отрицательным. Если оно отрицательно, то нужно изменить значение переменной S.

6. Вывод результата S.

IV. Описание переменных: m, n — `integer`, a, S — `real`.

19.4. Построение таблицы значений функции

Пример 19.9. Вывести на экран таблицу значений функции $y = \frac{x+2}{x^2+3}$. Количество значений вводится. Начальное значение $x = -3$, значения аргумента выводятся с шагом $h = 0{,}5$.

Этапы выполнения задания

I. Исходные данные: k (количество точек).

II. Результат: k значений аргумента и соответствующих им значений функции.

III. Алгоритм решения задачи.

1. Ввод числа k.

2. Для получения таблицы нужно в цикле вычислять и выводить значение

аргумента и соответствующее ему значение функции:

1) начальное значение аргумента $x = -3$. Для получения очередного значения аргумента нужно к текущему значению прибавить шаг h;

2) значение функции вычисляется по формуле $y = \frac{x+2}{x^2+3}$;

3) полученные значения выводятся на экран. Для того чтобы значения выводились строго одно под другим, нужно использовать форматный вывод. Для этого задать количество позиций для вывода значения и определить количество цифр после запятой. Запись x:7:2 означает, что для вывода переменной используется 7 позиций, после запятой выводятся 2 цифры.

3. Поскольку количество точек известно, воспользуемся циклом **for**.

IV. Описание переменных: k — integer, x, y, h — real.

19.5. Выделение цифр из числа

Пример 19.10. Дано натуральное число n. Вывести цифры числа по одной в строке (начиная с разряда единиц). Определить, сколько цифр в числе.

Этапы выполнения задания

I. Исходные данные: n (число).

II. Результат: z (текущая цифра числа), k (количество цифр в числе n).

III. Алгоритм решения задачи.

1. Ввод исходных данных — число n.

2. Определение начального значения счетчика для количества цифр ($k := 0$).

3. Количество цифр числа равно количеству десятичных разрядов в числе.

Понятие числа возникло в глубокой древности из практической потребности людей. Для записи чисел используют цифры. Любая информация в компьютере представляется с помощью всего двух цифр (0 и 1).

Числовой код имеет каждая страна мира, цифры задают пин-код банковской карты. Сегодня с помощью цифр можно получить числовой образ практически любого объекта.

Пример 19.10.

V. Программа:

```
var k,n,z: integer;
begin
  write('Введите n = ');
  read(n);
  k := 0;
  while n > 0 do
  begin
    //Текущая цифра
    z := n mod 10;
    writeln(z);
    //Уменьшение числа в 10 раз
    n := n div 10;
    //Подсчет кол-ва цифр
    k := k + 1;
  end;
  writeln('в числе ', k,
          'цифр(-а/-ы))');
end.
```

VI. Тестирование.

Запустить программу и ввести значение $n = 13579$. Результат:

Окно вывода

```
Введите n = 13579
9
7
5
3
1
В числе 5 цифр(-а/-ы)
```

Запустить программу и ввести значение $n = 1$. Результат:

Окно вывода

```
Введите n = 10
1
В числе 1 цифр(-а/-ы)
```

Алгоритм Евклида — алгоритм для нахождения наибольшего общего делителя двух целых чисел. Алгоритм назван в честь древнегреческого математика Евклида (III в. до н. э.), который впервые описал его в книгах «Начала». Это один из старейших численных алгоритмов, используемых в наше время.

Евклид использовал данный алгоритм не только для чисел, но и для геометрических величин: длин, площадей, объемов. В XIX в. алгоритм был обобщен на другие математические объекты. Сегодня алгоритм Евклида используется при шифровании данных.

Пример 19.11. Алгоритм Евклида для чисел 42 и 24:

№	a	b
1	42	24
2	18 (42–24)	24
3	18	6 (24–18)
4	12 (18–6)	6
5	6 (12–6)	6

Пример 19.12.

V. Программа:

```
var a,b:integer;
begin
  write('Значки у Иры a = ');
  read(a);
  write('Значки у Игоря b = ');
  read(b);
  while a<>b do
    if a>b then
      a := a - b
    else
      b := b - a;
  writeln('Всего друзей = ',a);
end.
```

VI. Тестирование.

Запустить программу и ввести значения: $a = 42$, $b = 24$. Результат:

Окно вывода

```
Значки у Иры a = 42
Значки у Игоря b = 24
Всего друзей = 6
```

Для нахождения каждой цифры числа нужно:

а) разделить число на 10;

б) найти целую часть от деления и остаток (остаток и будет очередной цифрой) и увеличить счетчик количества цифр;

в) вывести полученную цифру;

г) поскольку количество цифр в числе заранее неизвестно, будем использовать цикл **while**. Пока целая часть от деления больше 0, в числе еще есть цифры и нужно перейти к выполнению пункта а), иначе все цифры найдены.

4. Вывод значения переменной k.

IV. Описание переменных: k, n, z — integer.

19.6. Наибольший общий делитель двух чисел

Наибольшим общим делителем (НОД) для двух целых чисел называют наибольший из их общих делителей. Пример: для чисел 42 и 24 наибольший общий делитель равен 6.

Существуют несколько алгоритмов нахождения НОД. С одним из них вы знакомились на уроках математики. Нужно разложить каждое из чисел на простые множители, выбрать общие и перемножить.

Рассмотрим другой алгоритм, который называется алгоритм Евклида.

1. Из большего числа вычитаем меньшее.

2. Если получается 0, то числа равны друг другу и это значение является НОД.

3. Если результат вычитания не равен 0, то большее число заменяем на разность большего и меньшего.

4. Переходим к пункту 1.

(Рассмотрите пример 19.11.)

Пример 19.12. Ира и Игорь коллекционируют значки. У Иры в коллекции a значков, а у Игоря — b. Поскольку значков много, ребята решили поделиться своими значками с друзьями. Какое наибольшее количество общих друзей может быть у Иры и Игоря, если каждый из них хочет разделить все свои значки между друзьями без остатка? Например, если $a = 42$ и $b = 24$, то значки можно разделить, если у Иры и Игоря 1, 2, 3 или 6 общих друзей. Наибольшее количество — 6.

Этапы выполнения задания

I. Исходные данные: a и b (количество значков у Иры и у Игоря).

II. Результат: наибольшее количество общих друзей.

III. Алгоритм решения задачи.

1. Ввод чисел a, b.

2. Поскольку значки нужно делить без остатка, то ответом на задачу может быть только общий делитель чисел a и b. Среди всех делителей нужно найти наибольший.

3. Для решения задачи напишем программу вычисления НОД(a, b) по алгоритму Евклида. Пока числа a и b не равны, выполняем следующее:

1) сравниваем два числа;

2) если $a > b$, заменяем a на разность $a - b$, иначе заменяем b на разность $b - a$.

4. Вывод результата. Вывести можно как значение a, так и b (поскольку они равны).

IV. Описание переменных: a, b — `integer`.

Пример 19.13*. Написать программу вычисления НОД(x, y, z).

Пример 19.12. *Продолжение.*

Для значений $a = 1449$, $b = 596$ получим:

Окно вывода

```
Значки у Иры a = 1449
Значки у Игоря b = 596
Всего друзей = 1
```

Данный результат означает, что числа 1449 и 596 взаимно простые и все значки можно отдать только одному другу.

*В Pascal, кроме процедур, часто используются вспомогательные алгоритмы в виде функций. Функция всегда возвращает значение, которое нужно присвоить какой-либо переменной или использовать в любом выражении: `b := abs(x), t := 2 + sqrt(a)`.

Общий вид функции:

```
function <имя>(<список параметров>:тип): тип результата;
var ... //Может отсутствовать
begin
  <команды>
  <имя>:= <значение>;
end;
```

Пример 19.13*.

V. Программа:

```
var x,y,z,d,f:integer;
function NOD
         (a,b:integer):integer;
begin
  while a<>b do
    if a>b then
    a := a-b
    else
    b := b-a;
  NOD := a;
end;
begin
  write('Введите x = ');
  read(x);
  write('Введите y = ');
  read(y);
  write('Введите z = ');
  read(z);
  d := NOD(x,y);
  f := NOD(d,z);
  writeln('НОД = ',f);
end.
```

Пример 19.13*. *Продолжение.*

VI. Тестирование.

Запустить программу и ввести значения: $x = 26$, $y = 143$, $z = 65$. Результат:

```
Окно вывода
Введите x = 26
Введите y = 143
Введите z = 65
НОД = 13
```

Запустить программу и ввести значения: $x = 354$, $y = 847$, $z = 125$. Результат:

```
Окно вывода
Введите x = 354
Введите y = 847
Введите z = 125
НОД = 1
```

Данный результат означает, что числа 354, 847 и 125 взаимно простые.

Этапы выполнения задания

I. Исходные данные: x, y и z (три числа).

II. Результат: НОД (x, y, z).

III. Алгоритм решения задачи.

1. Ввод чисел x, y, z.

2. Используем то, что НОД (x, y, z) = = НОД (НОД (x, y), z). То есть сначала вычислим d = НОД (x, y), а затем f = НОД (d, z).

3. Для вычисления НОД двух чисел составим функцию NOD (a, b). Команды функции NOD рассмотрены в примере 19.12.

4. Вывод результата.

IV. Описание переменных: x, y, z, d, f — `integer`.

Упражнения

1 Выполните задания для примера 19.4.

1. Внесите в программу изменения так, чтобы числа выводились в обратном порядке — от большего к 2.

2. Какие изменения нужно внести в программу, чтобы вывести все четные числа меньше введенного числа x?

3. Внесите изменения в программу так, чтобы выводились числа, кратные 3, 5; кратные введенному числу x.

4. Измените программу так, чтобы можно было вывести все числа последовательности, заданной формулой $a_n = \frac{n}{n^2+1}$ (пример 19.1).

2 Выполните задания для примера 19.5.

1. Запустите программу для разных значений x.

2. Какое максимальное значение x можно ввести?

3. Замените в программе тип `integer` на тип `int64`. Сколько чисел последовательности Фибоначчи можно найти теперь?

4. Измените программу так, чтобы она выводила значение числа Фибоначчи по введенному номеру.

3 Выполните задания для примера 19.6.

1. Запустите программу несколько раз для одного и того же значения (например, 20). Какое число получается в ответе чаще всего? Почему?

2. Измените диапазон случайных чисел на [1, 1000] и проверьте, какое число получается в результате чаще других для тех же 20 элементов.

3. Измените программу так, чтобы вычислялось количество чисел, кратных введенному числу x.

4. Измените программу так, чтобы можно было посчитать количество чисел из промежутка $[a; b]$, a и b вводятся.

4* Напишите программу, которая будет выводить на экран элементы последовательности трибоначчи — первые элементы последовательности: 0, 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, ... Каждый элемент, начиная с четвертого, равен сумме трех предыдущих: $a_n = a_{n-1} + a_{n-2} + a_{n-3}$.

1. По заданному n вывести элемент последовательности.

2. Для заданного x вывести элементы последовательности меньше x.

5 Выполните задания для примера 19.7.

1. Замените в решении задачи цикл **for** на цикл **while**.

2. Выполните программу для $m = 2000$. Почему в ответе сумма = NaN? Какие изменения нужно внести в программу для получения результата? Найдите значение для $m = 2000$, $m = 10\,000$.

6 Найдите сумму первых m элементов последовательности. Число m вводится. Элементы последовательности задаются формулой $a_n = \frac{1}{n^3}$.

1. На сколько большим должно быть значение m, чтобы программа выдала ответ «сумма = Infinity»? Почему так произошло?

2. Какие изменения нужно внести в программу для получения правильного результата?

3. Замените в решении задачи цикл **for** на цикл **while**. Найдите сумму для $m = 20\,000$, $m = 1\,000\,000$.

7 Выполните задание для примера 19.8. Измените форму вывода результата таким образом, чтобы результат выводился в виде a ^ n = S (например, для значений $a = 2$, $n = 3$ должно быть напечатано 2 ^ 3 = 8).

8 Факториалом числа n называют произведение всех натуральных чисел, не превосходящих n. Обозначают факториал так: $n!$ По определению факториал числа 0 равен 1. Напишите программу, которая вычислит значение факториала целого неотрицательного числа n. Для проверки можно использовать следующее: 0! = 1; 2! = 2; 5! = 120, 10! = 3 628 800.

9 Выполните задание для примера 19.9. Замените в решении задачи цикл **for** на цикл **while**. В качестве условия в цикле **while** можно использовать следующее: $x <= 3$.

10 Постройте таблицы значений для указанных функций.

1. $y = x^2 - 5x - 3$, $x \in [-3, 3]$, вводится значение шага h.

2. $y = 2 + \frac{3x^3 - 7}{x}$, $x \in [a, b]$[1], вводятся значения a, b и количество точек.

[1] Подсказка: $h = \frac{a+b}{k-1}$.

11 Выполните задания для примера 19.10.

1. Команду `writeln('в числе ', k, ' цифр')` заменили командой `writeln('число ',n,' состоит из ', k, ' цифр')`. Какой результат будет получен и почему? Какие изменения нужно внести в программу для получения правильного результата?
2. Изменится ли результат работы программы, если вместо условия цикла $n < 0$ использовать условие $n > 1$?
3. Проверьте работу программы для $n = 0$. Почему получился такой результат? Что нужно изменить в программе для получения правильного результата?

12 Программу из примера 19.10 изменили. Сформулируйте задачу, которая решается с помощью данной программы.

```
var i,k,n,z: integer;
begin
 write('Введите n = ');
 read(n);
 write('Введите i = ');
 read(i);
 k:=0;
while n > 0 do
begin
 z : = n mod 10; //Текущая цифра
 k : = k + 1;
 if k = i then
  writeln('В разряде ', i, ' стоит цифра ', z);
 N : = n div 10; //Уменьшение числа в 10 раз
end;
if i > k then
 writeln('В числе ', k, ' цифр, в разряде ', i, ' нет цифр')
else
 writeln('В числе ', k, ' цифр');
end.
```

13 Дано натуральное число n. Определите, каких цифр в числе больше — четных или нечетных.

14 Дано натуральное число n. Выведите номера разрядов, в которых стоят цифры, кратные 3, или сообщение, что таких цифр нет.

15 Задано натуральное число. Напишите программу, которая для числа с нечетным количеством цифр выведет на экран цифру, стоящую на средней позиции числа.

Если в числе четное количество цифр, то вывести соответствующее сообщение. Например, для числа 23 452 ответом будет 4, а для числа 56 — сообщение «В числе четное количество цифр».

16 Число является палиндромом, если оно одинаково читается слева направо и справа налево. Напишите программу, которая по введенному натуральному числу определит, является оно палиндромом или нет. Примеры: 12321, 6776 — палиндромы, 12335 — нет.

17* Натуральное число n называется числом Армстронга, если сумма цифр числа, возведенных в n-ю степень, где n — количество цифр в числе, равна самому числу. Например, $153 = 1^3 + 5^3 + 3^3$. Напишите программу, которая найдет:

1. Все трехзначные числа Армстронга.
2. Все четырехзначные числа Армстронга.

18 В 1626 г. индейцы продали остров Манхэттен за 20 долларов. Если бы эти деньги были помещены в банк на текущий счет и ежегодный прирост составлял бы x %, какова была бы стоимость капитала в этом году? Напишите программу, которая ответит на данный вопрос.

19 Имеются два сосуда. В первом находится C_1 л воды, а во втором C_2 л воды. Из первого сосуда переливают половину воды во второй, а затем из второго переливают половину воды в первый (одно переливание) и т. д. Напишите программу, которая определит, сколько воды окажется в каждом из сосудов после k переливаний.

20 Дано натуральное число n. Написать программу, которая выведет все числа, взаимно простые с n, а также числа, которые меньше его.

21* Измените программу из примера 19.13:

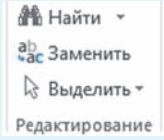
1. Найти НОД четырех чисел.
2. Найти НОК (наименьшее общее кратное) двух чисел.
3. Вводятся числитель и знаменатель правильной дроби. Сократите дробь.

4*. Две правильные дроби заданы своими числителями и знаменателями. Найдите их сумму. Ответ выведите в виде смешанной дроби.

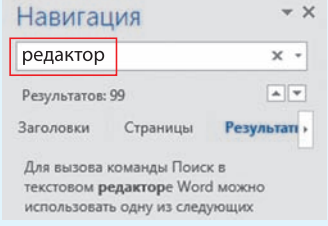
Глава 4
ТЕХНОЛОГИЯ ОБРАБОТКИ ТЕКСТОВЫХ ДОКУМЕНТОВ

§ 20. Редактирование текста

Пример 20.1. Группа **Редактирование.**



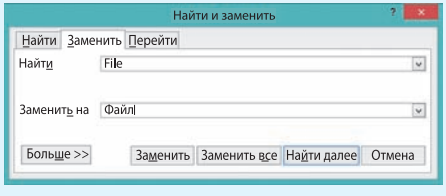
Пример 20.2. Панель **Навигация.**



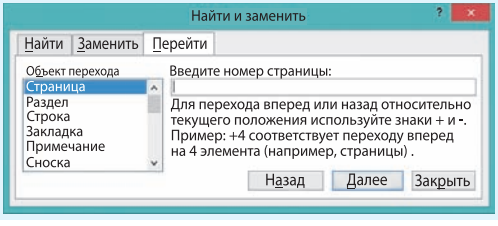
Найденные слова выделены цветом:

Для вызова команды **Поиск** в текстовом редакторе Word можно использовать одну из следующих возможностей:

Пример 20.3. Вкладка **Заменить** в окне **Найти и заменить.**



Вкладка **Перейти** окна **Найти и заменить** содержит команды для навигации по структурным элементам текстового документа.



20.1. Поиск и замена в тексте

Текстовые редакторы позволяют осуществлять в тексте поиск слов или словосочетаний и при необходимости заменять их другими словами, словосочетаниями.

Для вызова команды **Поиск** в текстовом редакторе Word можно использовать одну из следующих возможностей:

- команду **Найти** на вкладке **Главная** в группе **Редактирование** (пример 20.1);
- комбинацию клавиш Ctrl + F.

После вызова команды **Поиск** необходимо ввести образец для поиска в соответствующем поле панели **Навигация** (пример 20.2).

Для вызова команды **Заменить** используется одна из следующих возможностей:

- команда **Заменить** на вкладке **Главная** в группе **Редактирование** (см. пример 20.1);
- комбинация клавиш Ctrl + H.

После того как выбрана команда **Заменить**, необходимо заполнить образец поиска в поле **Найти** (что ищем) и образец замены в поле **Заменить** (на что заменяем). Вид окна при замене текста показан в примере 20.3.

В *Приложении 4* (с. 163) можно ознакомиться с другими возможностями поиска и замены.

20.2. Проверка правописания

При создании текстового документа возможны ошибки. Современные текстовые редакторы имеют встроенные системы проверки правописания.

Система проверки правописания — компьютерная программа, осуществляющая проверку заданного текста на наличие в нем орфографических ошибок.

Текстовый редактор Word осуществляет поиск ошибок с помощью встроенных словарей (пример 20.4). Если слово из текста отсутствует в словаре, то оно подчеркивается волнистой красной линией. Части текста, в которых допущена стилистическая ошибка или ошибка форматирования текста, подчеркиваются волнистой голубой или зеленой линией.

Возможные правильные варианты написания слова или расстановки знаков препинания можно узнать из контекстного меню подчеркнутого слова (пример 20.5). Аналогичный результат можно получить, если щелкнуть по значку  , который расположен в строке состояния. Справа откроется панель, в которой показаны варианты исправления ошибки (пример 20.6) или объяснение того, почему текст подчеркнут голубой (зеленой) волнистой линией (пример 20.7).

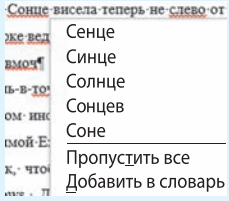
Вспомним некоторые правила ввода компьютерного текста:

1. Разделителем между двумя словами является **один** пробел. Большее количество пробелов воспринимается как ошибка и подчеркивается голубой волнистой линией.

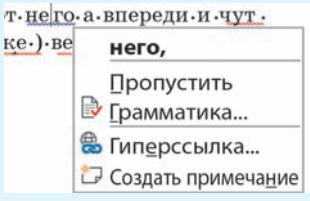
Пример 20.4. Выделение ошибок в Word.

Но · влева · некакой · дороги ·· не · было. · Сонце · висела · теперь · не · слево · от · него · а · впереди · и · чут · справо. · Вправа · (· на · бугарке ·) · веднелся · хутор.¶

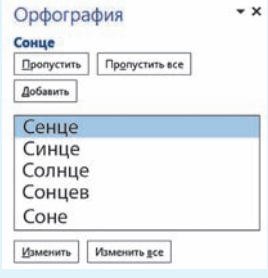
Пример 20.5. Варианты правильного написания слова в контекстном меню:



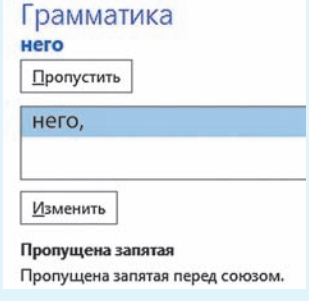
Расстановка знаков препинания:



Пример 20.6. Панель **Орфография**.



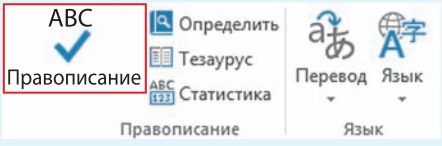
Пример 20.7. Панель **Грамматика**.



Системы проверки правописания на персональных компьютерах появились в 1980 г. и были автономными программами. В 1980-х гг. их включили в состав программ для работы с текстом.

Сегодня проверка правописания существует не только для текстовых редакторов, но и для веб-браузеров.

Пример 20.8. Команды орфографического контроля на вкладке **Рецензирование**.

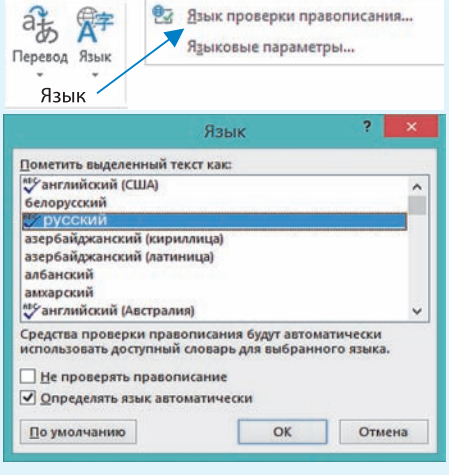


В Word можно искать синонимы или антонимы для выделенного слова (команда **Рецензирование** → **Тезаурус**).

Для расстановки переносов в тексте используется команда **Разметка страницы** → **Расстановка переносов**.

Для языков, для которых осуществляется орфографический контроль, возможен перевод текста или отдельных слов. Для этого используется команда **Рецензирование** → **Перевод**.

Пример 20.9. Окно **Язык**.



2. После знаков препинания («.» «,» «:» «;» «!» «?») обязательно должен быть пробел (но не перед ними). Если после знака препинания пробела нет, а сразу записано новое слово, то Word воспринимает эти два слова как одно и подчеркивает, как орфографическую ошибку.

3. Тире отделяется пробелами с двух сторон. Дефис пробелами не отделяется. Для постановки длинного тире (—), а не короткого (–) можно воспользоваться комбинацией клавиш Ctrl + «–» (на цифровой клавиатуре).

4. Пробел ставится перед открытой скобкой и после закрытой. После открытой скобки и перед закрытой пробел не ставится. Это же правило применяется при использовании кавычек.

Если автоматическая проверка орфографии отключена, найти ошибки в тексте можно, запустив проверку правописания командой **Правописание** вкладки **Рецензирование** (пример 20.8) или клавишей **F7**.

Орфографический контроль осуществляется не для всех доступных языков. Список доступных языков можно посмотреть, выбрав команду **Язык проверки правописания** из выпадающего списка команды **Язык** (пример 20.9). Если рядом с названием языка стоит значок  , то для данного языка возможен орфографический контроль. Из примера 20.9 видно, что для русского и английского языков орфографический контроль осуществляется, а для белорусского языка — нет.

1. Как осуществляется поиск в тексте?
2. Как заменить одно слово другим?
3. Что понимают под системой проверки правописания?
4. Как Word помечает ошибки в тексте?
5. Какие правила необходимо соблюдать при вводе компьютерного текста?

Упражнения

1 Откройте текстовый документ. Исправьте ошибки, используя возможности орфографического контроля. Объясните, почему не подчеркиваются красными линиями некоторые слова, написанные неверно.

1. Но влева некакой дороги не было. Сонце висела теперь не слево от него а впереди и чут справо. Вправа на бугарке веднелся хутор.
2. В тексте можна асставляе знаки периносов автаматически или вручную. При ручном воде Word ищит слова которые возможно перинести и запрашывает разришение вставки переноса. При автомаческом введении переносов Word сам асставляет знаки там, где эта необходимо.

Правильный вариант

1. Но влево никакой дороги не было. Солнце висело теперь не слева от него, а впереди и чуть справа. Справа на бугорке виднелся хутор[1].
2. В тексте можно расставить знаки переносов автоматически или вручную. При ручном вводе Word ищет слова, которые возможно перенести, и запрашивает разрешение вставки переноса. При автоматическом введении переносов Word сам расставляет знаки там, где это необходимо.

2 Откройте текстовый документ. При наборе текста на русском языке случайно включили белорусскую раскладку клавиатуры. Используя функцию **Замена**, исправьте текст.

Інструкція по поіску сінонімов і антонімов

Для начала небольшая справка (с сайта Грамота.ру).

Сінонім — слово, отлічаюўееся от другого по звучанію ілі напісанію, но совпадаюўее ілі блізкое ему по значенію.

Антонім — слово с протівоположным по отношенію к другому слову значеніем.

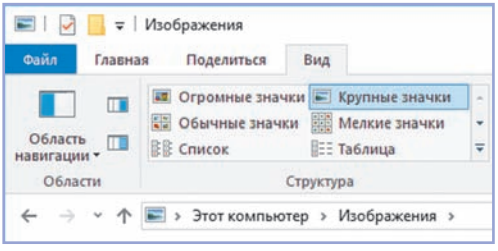
Тезаурус — словарь какого-лібо языка, представляюўій его лексіку в полном об'еме.

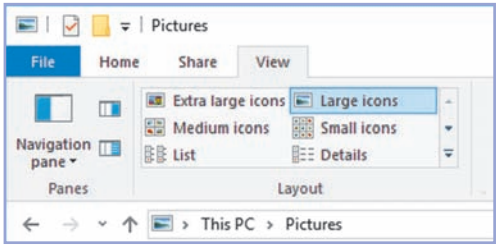
Выбор сінонімов і/ілі антонімов для определенного слова осуўествляется ўелчком правой кнопкі мыші на слове і наведеніем указателя мыші на команду **Сінонімы** (ілі командой **Тезаурус**).

Прі выборе команды **Тезаурус** в редакторе Word открывается панель **Тезаурус**, в которой будут разліčные варіанты подходяўіх слов.

[1] Цитируется по произведению А. Гайдара «Пусть светит». Режим доступа: http://modernlib.net/books/gaydar_arkadiy/pust_svetit/read. Дата доступа: 10.02.2018.

3 Откройте текстовый документ с инструкцией по работе с программой **Explorer** (**Проводник**) для англоязычной версии Windows 10. Исправьте инструкцию так, чтобы она позволила пользователю работать с русской версией Windows.



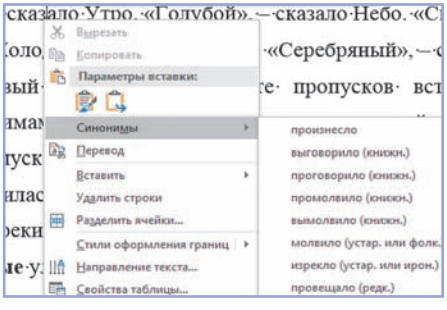


Для этого с помощью команды **Заменить** выполните следующие замены во всем тексте сразу: *Home — Главная*, *View — Вид*, *Icons — значки*. Замените: *Explorer — Проводник*, *New folder — Создать папку*, *Delete — Удалить*, *Move to — Переместить в*, *Copy to — Копировать в*, *Extra Large — огромные*, *Large — крупные*, *Medium — обычные*, *Small — мелкие*.

Краткая инструкция по работе с программой Explorer

1. Создать папку: **Home → New Folder.**
2. Удалить: **Home → Delete.**
3. Переместить: **Home → Move to.**
4. Копировать файл: **Home → Copy to.**
5. Представление файлов в виде огромных значков: **View → Extra Large Icons.**
6. Представление файлов в виде крупных значков: **View → Large Icons.**
7. Представление файлов в виде обычных значков: **View → Medium Icons.**
8. Представление файлов в виде мелких значков: **View → Small Icons.**

4* Загрузите текстовый документ. Замените слово *сказал*, встречающееся в тексте, его синонимами.



Заспорили пуночки (северные воробьи), не могут решить, какой бывает снег. «Золотой», — сказало Утро. «Голубой», — сказало Небо. «Синий-синий», — сказали Тени. «Холодный», — сказала Утка. «Серебряный», — сказала Луна.

5* Загрузите текстовый документ. На месте пропусков вставьте слова, являющиеся антонимами к выделенным. Указание: сначала скопируйте выделенное слово на место пропуска, а затем замените его антонимом.

Погода испортилась: **ясные** и **теплые** дни сменились ... и ... .
Правый берег реки был **пологий**, а левый ... .
Узкие и **кривые** улицы старой Москвы сменились ... и ... .
Из **тесных, темных** бараков рабочие переселились в ... и ... квартиры.
В конце долгого пути даже **легкий** труд становится ... .

§ 21. Списки и колонки

21. 1. Создание и форматирование списков

При создании текста некоторые абзацы приходится нумеровать (пример 21.1) или выделять при помощи различных маркеров (пример 21.2).

Абзацы, отмеченные маркерами или номерами, образуют **список.**

Список абзацев, отмеченных номерами, называют **нумерованным.** Список абзацев, отмеченных маркерами, называют **маркированным.**

Для создания списков можно использовать кнопки на вкладке **Главная:**

 — маркированный список. Элементы такого списка начинаются со строчной буквы и заканчиваются запятой или точкой с запятой;

 — нумерованный список (с точкой после номера). Элементы списка начинаются с заглавной буквы и заканчиваются точкой.

Эти же команды присутствуют в контекстном меню абзаца (пример 21.3).

Для оформления готового текста в виде списка нужно выделить те абзацы, которые будут образовывать

Пример 21.1. Нумерованные списки.

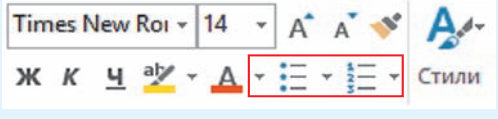
Расписание уроков на понедельник:
1. Английский язык.
2. Математика.
3. Физкультура.
4. Химия.
5. Русская литература.
6. История.

Какие типы эффектов анимации реализованы в программе Power Point? Выберите правильные ответы:
а) эффекты выхода;
б) эффекты входа;
в) эффекты разделения;
г) эффекты размещения;
д) эффекты выделения;
е) эффекты изменения.

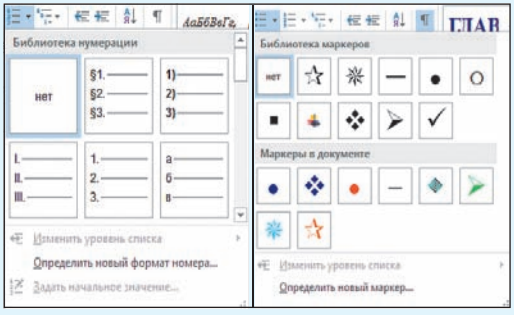
Пример 21.2. Маркированный список.

Сегодня в школьной столовой пирожки с начинкой из:
- капусты;
- картошки;
- рыбы;
- творога;
- кураги.

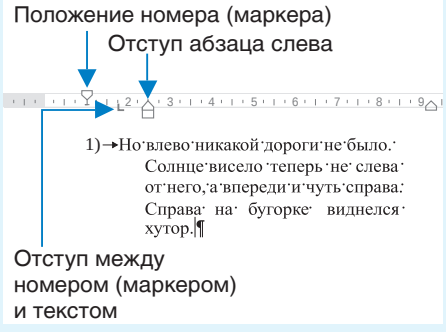
Пример 21.3. Команды для создания списков в контекстном меню.



Пример 21.4. Выбор формата номера и маркера для списка.



Пример 21.5. Маркеры на линейке (включен режим отображения непечатных символов).



Пример 21.6. Многоуровневый

Времена года

1. ЗИМА:
   - декабрь;
   - январь;
   - февраль.
2. ВЕСНА:
   - март;
   - апрель;
   - май.
3. ЛЕТО:
   - июнь;
   - июль;
   - август.
4. ОСЕНЬ:
   - сентябрь;
   - октябрь;
   - ноябрь.

Пример 21.7. Многоуровневый — часть содержания учебного пособия.

список, а затем выбрать вид списка. Подробнее о создании списков и о правилах их оформления можно прочитать в *Приложении 4* (с. 163—165).

Нумеровать абзацы можно арабскими или римскими цифрами, русскими или латинскими буквами. Различные маркеры для списков можно выбрать из библиотеки или использовать в качестве маркеров рисунки (пример 21.4). Номер или маркер можно форматировать, как и символы: изменить шрифт, размер или цвет шрифта, применить начертание.

Если при вводе текста первым символом набрать число, то после нажатия клавиши Enter этот абзац будет автоматически оформлен в виде списка. Следующий абзац получит номер на 1 больше. Управлять размещением списка на странице можно с помощью маркеров на линейке (пример 21.5).

Текстовый редактор Word позволяет создавать списки сложной структуры.

Списки, в которых применяются одновременно различные виды номеров и/или маркеров, называют **многоуровневыми.**

В многоуровневых списках каждый уровень имеет свою нумерацию (пример 21.6). Номер нового уровня может включать в себя номер предыдущего уровня, который обычно отделяется точкой. Примером такого списка является оглавление учебного пособия (пример 21.7).

21.2. Колонки в текстовом документе

Word позволяет размещать текст на странице как в газетных колонках, в которых текст непрерывно перетекает из нижней части одной колонки в верхнюю часть следующей (пример 21.8). С помощью колонок можно оформлять рекламные проспекты, буклеты, брошюры.

Для создания колонок набранный текст выделяют и выполняют команду **Колонки,** которая находится на вкладке **Разметка страницы** (пример 21.9). Число строк в созданных колонках будет примерно одинаковым (возможное исключение — последняя колонка).

Команда **Другие колонки** открывает окно **Колонны**, где можно настраивать внешний вид колонок (пример 21.10). Максимальное количество колонок зависит от ширины листа (для стандартного листа А4 — 12). Чтобы сделать колонки различной ширины, необходимо убрать птичку в поле **Столбцы одинаковой ширины**. Поле **Разделитель** позволяет вставить между колонками разделительную линию.

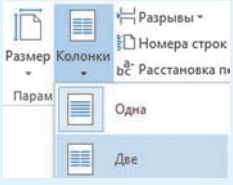
Если текст при наборе оформляют в виде колонок, то в конце первой колонки вставляют разрыв колонки: команда **Колонка** в выпадающем списке **Разрывы** на вкладке **Разметка страницы** (пример 21.11). Команде соответствует комбинация клавиш Ctrl + Shift + Enter. Данная команда вставляет специальный символ — **«Разрыв столбца»**, который показывает конец текущей колонки. Он отображается в режиме показа непечатаемых символов (пример 21.12).

Пример 21.8. Колонки в тексте.

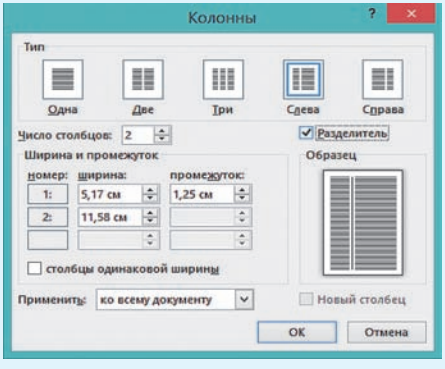
Помимо курсов, дисциплина станет обязательной частью программы учреждений образования, что существенно повысит ИТ-грамотность населения.

Главной целью ближайшего будущего будет научить писателей, биологов и бухгалтеров использовать компьютеры для решения своих задач. Например, врач, который использует компьютерные программы для более точной постановки диагноза, будет больше соответствовать реалиям будущего.

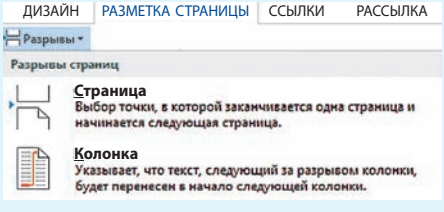
Пример 21.9. Создание колонок.



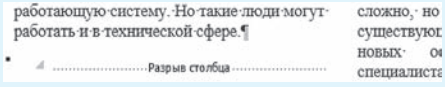
Пример 21.10. Окно с параметрами управления колонками.



Пример 21.11. Вставка разрыва колонки.



Пример 21.12. Принудительный конец колонки.



1. Что такое список?
2. Какие существуют виды списков?
3. Как создать список?
4. Как оформить текст в виде колонок?

Упражнения

1 Наберите текст (или откройте текстовый документ). Оформите маркированный список и примените нужное форматирование.

Музыканты-романисты черпали вдохновение в народных песенных мотивах и танцевальных ритмах. Часто обращались в своем творчестве к литературным произведениям Шекспира, Гёте, Шиллера. В XIX в. многие европейские страны дали миру великих композиторов:

- *Австрия* и *Германия* — **Франц Шуберт** и **Рихард Вагнер;**
- *Польша* — **Фредерик Шопен;**
- *Венгрия* — **Ференц Лист;**
- *Италия* — **Джоакино Россини** и **Джузеппе Верди;**
- *Чехия* — **Бедржих Сметана;**
- *Норвегия* — **Эдвард Григ;**
- *Россия* — **Михаил Глинка, Александр Бородин, Петр Чайковский.**

2 Наберите тексты на одном из иностранных языков (или откройте текстовый документ). Оформите нумерованный список и примените нужное форматирование к тексту.

Текст на английском языке:

Colour Idioms

1) To give a **black** look — гневно взглянуть;
2) once in a **blue** moon — очень редко, почти никогда;
3) to be (feel) **blue** — грустить, быть в плохом настроении;
4) to be like a **red** rag to a bull — действовать на кого-либо, как красная тряпка на быка;
5) a **white** elephant — дорогой, но бесполезный подарок;
6) to be ***yellow*** — струсить, быть трусом.

Текст на французском языке:

Des mots en couleurs

1) Mettre dans le **noir** — попасть в самую точку;
2) passer du **blanc** au noir — переходить из одной крайности в другую;
3) couper le bois à **blanc** — вырубить весь лес;
4) avoir une peur **bleue** — ужасно испугаться;
5) en dire de ***vertes*** — позволить себе вольности в разговоре;
6) faire *gris* mine à qu — встретить кого-либо с кислой миной.

Текст на испанском языке:

Colores en expresiones

1) suerte **negra** — горькая доля;
2) no distinguir lo blanco de lo **negro** — совершенно не разбираться в чем-либо;
3) dejar en **blanco** — обвести вокруг пальца;
4) al **rojo** **blanco** — до белого каления;
5) ponerse **rojo** — покраснеть от стыда;
6) el que quiera **azul** celeste, que le cueste — любишь кататься — люби и саночки возить.

Текст на немецком языке:

Die geflügelte bunte Worte

1) die **schwarze** Kunst — типографское дело;
2) **blauen** Montag machen — прогулять;
3) es wurde **grün** und **blau** vor den Augen — потемнело в глазах;
4) sich ***gelb*** und ***grün*** ärgern — быть вне себя;
5) **gelbe** Neid — черная зависть;
6) das wirkt auf him wie ein **rotes** Tuch — это действует на него, как красная тряпка на быка.

3 Откройте текстовый документ «Какой может быть работа ИТ-специалиста будущего». Оформите текст в три колонки в соответствии с образцом. Каждая колонка должна начинаться с подзаголовка (текст выделен полужирным начертанием).

Какой может быть работа ИТ-специалиста будущего[1]

Существуют разные прогнозы, но одно ясно точно — компьютерные науки никуда не денутся. По какому бы сценарию ни развивался мир, такие навыки можно будет применить во многих профессиях и в будущем.

Доступное программирование

На популяризацию программирования среди детей и взрослых направлено множество проектов.

Главная цель ближайшего будущего — научить представителей разных профессий. Например, врач, который применяет компьютерные программы для точной постановки диагноза, или менеджер по продажам, который может фильтровать данные клиентов для более эффективной работы, будут больше соответствовать реалиям будущего.

В коде только программисты

Число программистов во всем мире достигло 15 млн. Сейчас активно развиваются детские курсы программирования, где учат писать код на языке Scratch. Поэтому количество специалистов в будущем будет только расти.

Сегодня появляется все больше новых языков программирования. Они для восполнения пробелов в применении уже существующих языков. Поэтому потребность в программистах на сегодняшний день довольно высока.

Специальное программирование

Можно выделить сферы, в которых в ближайшем будущем потребуются в ИТ-специалисты.

Уже сегодня развивается такое направление, как биржевой анализ. Специалист данного профиля по сути является программистом в финансовой сфере.

Все большую популярность приобретает интернет вещей. Еще одно перспективное направление — искусственный интеллект и робототехника. Также развиваются облачные технологии.

[1] По материалам сайта https://nabr.com/company/1cloud/blog/316930/ (дата доступа 20.01.2018).

4* Наберите текст (или откройте документ). Создайте копию набранного текста. Распределите текст по двум колонкам. Оформите списки согласно образцу.

- Системное ПО
  - Операционные системы
  - Файловые менеджеры
  - Сетевые программы
  - Драйверы
  - Утилиты
- Прикладное ПО
  - Программы общего назначения
  - Программы специального назначения
  - Системы обучения
  - Компьютерные игры
- Инструментальное ПО
  - Системы программирования
  - Трансляторы
  - Отладчики
  - Наборы библиотек

Результат:

I. **Системное ПО**:
- ➢ операционные системы;
- ➢ файловые менеджеры;
- ➢ сетевые программы;
- ➢ драйверы;
- ➢ утилиты.

II. **Прикладное ПО**:
- ❋ программы общего назначения;
- ❋ программы специального назначения;
- ❋ системы обучения;
- ❋ компьютерные игры;

III. **Инструментальное ПО**:
- ☆ системы программирования;
- ☆ трансляторы;
- ☆ отладчики;
- ☆ наборы библиотек.

1. **Системное ПО**:
   1.1. Операционные системы.
   1.2. Файловые менеджеры.
   1.3. Сетевые программы.
   1.4. Драйверы.
   1.5. Утилиты.
2. **Прикладное ПО**:
   2.1. Программы общего назначения.
   2.2. Программы специального назначения.
   2.3. Системы обучения.
   2.4. Компьютерные игры.
3. **Инструментальное ПО**:
   3.1. Системы программирования.
   3.2. Трансляторы.
   3.3. Отладчики.
   3.4. Наборы библиотек.

§ 22. Таблицы

22.1. Создание таблиц

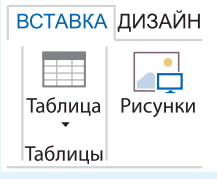
Современные текстовые редакторы позволяют вставлять в текст документа различные объекты: таблицы, рисунки, диаграммы, формулы и др. Большинство этих объектов можно добавить в текстовый документ, используя команды вкладки **Вставка**.

Таблицы позволяют структурировать текст, представить его более наглядно.

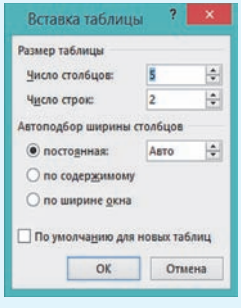
Вставить таблицу в документ можно с помощью команды **Таблица** на вкладке **Вставка** (пример 22.1). После нажатия на кнопку нужно выделить мышью область, определяющую количество строк и столбцов таблицы. Количество строк и столбцов можно задать числами, если выбрать команду **Вставить таблицу** (пример 22.2).

Многие действия, которые можно осуществлять с таблицей, определены

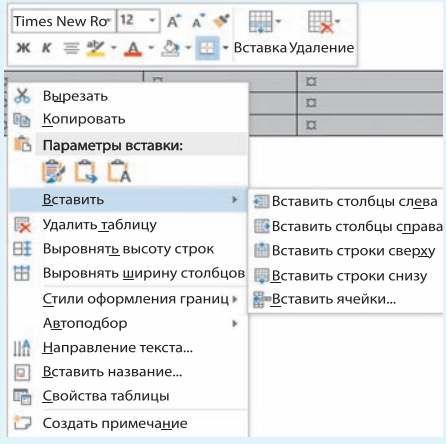
Пример 22.1. Вставка таблицы.



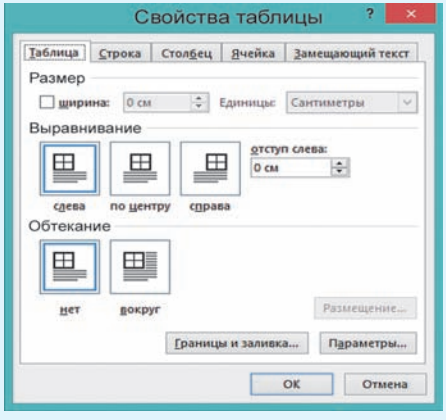
Пример 22.2. Определение размеров таблицы.



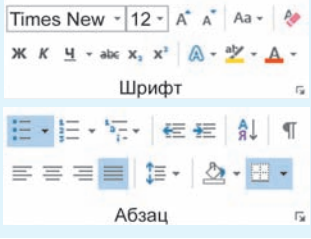
Пример 22.3. Контекстное меню таблицы.



Пример 22.4. Окно **Свойства таблицы.**



Пример 22.5. Команды для изменения абзацных и символьных свойств текста ячейки таблицы (вкладка **Главная**).



в ее контекстном меню (пример 22.3). Контекстное меню может содержать и другие команды, это зависит от того, какие элементы таблицы выделены. В контекстном меню можно выбрать, просмотреть и изменить свойства таблицы (пример 22.4). Данное окно позволяет задать расположение таблицы на странице документа: размер, выравнивание, отступ, обтекание. Вкладки **Строка**, **Столбец** и **Ячейка** позволяют управлять размерами соответствующих структурных элементов таблицы.

22.2. Форматирование таблиц

К форматированию таблицы относят изменение:

- абзацных и символьных свойств текста внутри ячейки;
- направления текста;
- ширины столбца или высоты строки;
- внешнего вида границ и фона ячейки;
- структуры таблицы: объединение ячеек, удаление и добавление строк или столбцов.

(Рассмотрите примеры 22.5.—22.8.)

Некоторые из команд форматирования таблицы размещены на вкладке **Главная**. Дополнительно для форматирования таблицы используются команды, размещенные на вкладках **Работа с таблицами → Конструктор** и **Работа с таблицами → Макет** (см. *Приложение 4*, с. 166).

К таблице размером 8 × 3 (пример 22.9, *а* применили следующее форматирование:

1. В первой строке выделили все ячейки и объединили их. Получили

ячейку для ввода заголовка таблицы. К заголовку применили следующее форматирование: шрифт Arial, размер 14, полужирный. Выравнивание по центру. Вокруг ячейки, содержащей название таблицы, убрали границы сверху, слева и справа.

2. Объединили ячейки в третьей и шестой строках. Закрасили фон ячеек. К тексту применили шрифт Times New Roman, размер 12, курсивное полужирное начертание.

3. Ввели текст в остальные ячейки таблицы, шрифт Times New Roman, размер 12.

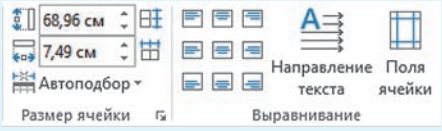
4. Для всех ячеек таблицы, содержащих числа, применили выравнивание по центру.

В примере 22.9, *б* к этой же таблице применили один из встроенных стилей таблиц.

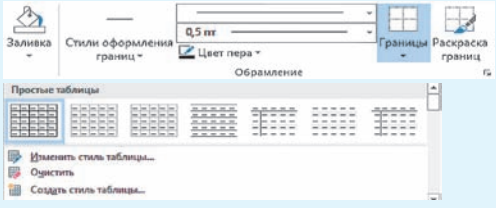
Иногда после создания таблицы приходится изменять ширину столбцов или высоту строк. Для изменения ширины столбца (высоты строки) необходимо навести указатель мыши на границу столбца (строки). Указатель мыши примет вид: ↔ (↕). Далее, удерживая нажатой левую клавишу мыши, нужно изменить размер строки (или столбца).

Для удаления столбца (строки) из таблицы необходимо его (ее) выделить. После этого выбрать команду **Удаление** (пример 22.10). Для выделенных столбцов (строк) команду **Удалять столбцы (строки)** можно выбрать из контекстного

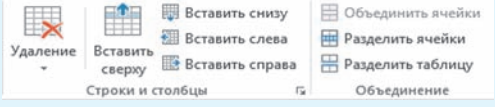
Пример 22.6. Команды для изменения направления текста, ширины столбца или высоты строки в таблице, выравнивания в ячейке (вкладка **Работа с таблицами → Макет**).



Пример 22.7. Команды для изменения внешнего вида границ и фона ячейки (вкладка **Работа с таблицами → → Конструктор**).



Пример 22.8. Команды для изменения структуры таблицы (вкладка **Работа с таблицами → Макет**).



Пример 22.9. Примеры таблиц.

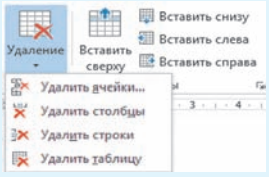
а

Плотность веществ		
Вещества	кг/м^3	г/см^3
Вещества в твердом состоянии		
Серебро	10 500	10,5
Фарфор	2300	2,3
Вещества в жидком состоянии		
Ртуть	13 600	13,60
Вода	1000	1,00

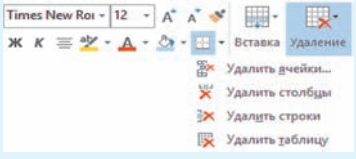
б

Плотность веществ		
Вещества	кг/м^3	**г/см^3**
Вещества в твердом состоянии		
Серебро	10 500	**10,5**
Фарфор	2300	**2,3**
Вещества в жидком состоянии		
Ртуть	13 600	**13,60**
Вода	1000	**1,00**

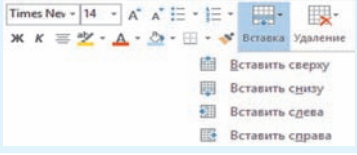
Пример 22.10. Команды для удаления (вставки) строк (столбцов) таблицы. Вкладка **Работа с таблицами** → → **Макет.**



Пример 22.11. Команды для удаления строк (столбцов) таблицы. Контекстное меню.



Пример 22.12. Команды для вставки строк (столбцов) из контекстного меню.



меню таблицы (пример 22.11). Если просто нажать клавишу Del, то удалятся только данные из выделенных ячеек таблицы.

Для вставки строк (столбцов) можно воспользоваться командами для вставки (пример 22.10). Вставляется столько строк (см. столбцов), сколько их выделено. Для вставки строк или столбцов можно воспользоваться и контекстным меню таблицы (пример 22.12). Если нажать клавишу Enter в конце какой-либо строки таблицы (за границей последней ячейки в строке), то после нее будет добавлена новая строка таблицы.

Используя команды вкладки **Работа с таблицами** → **Макет,** можно менять направление ввода текста, расположение текста относительно границ ячейки, задавать поля ячейки. При необходимости таблицу можно преобразовать в текст.

1. Как вставить таблицу в документ?
2. Что понимают под форматированием таблицы?
3. Как изменить высоту строки? Как изменить ширину столбца?
4. Как удалить строку (столбец) таблицы?
5. Как вставить строку (столбец) в таблицу?

Упражнения

1 Создайте одну из таблиц. Тексты стихов можно загрузить из файла. Примените к таблице форматирование по своему выбору.

Басня И. Крылова[1]	Перевод на **белорусский** язык Я. Купалы[2]
ЛЕБЕДЬ, ЩУКА И РАК Когда в товарищах согласья нет, На лад их дело не пойдет, И выйдет из него не дело, только мука.	**ЛЕБЕДЗЬ, ШЧУПАК І РАК** Там, дзе еднасці і згоды Няма шчырых у людзей, Праца іхняя заўсёды Ліха марна прападзе.

[1] Крылов, И. А. Басни; повести / [Сост. и предисл. П. Ткачева]. — Минск: БГУ, 1980.
[2] http://yankakupala.ru/lebedz-shchupak-i-rak (дата доступа 11.01.2018).

Басня И. Крылова	Перевод на **белорусский** язык Я. Купалы
Однажды Лебедь, Рак да Щука Везти с поклажей воз взялись, И вместе трое все в него впряглись: Из кожи лезут вон, а возу все нет ходу! Поклажа бы для них казалась и легка: Да Лебедь рвется в облака, Рак пятится назад, Щука тянет в воду. Кто виноват из них, кто прав — судить не нам; Да воз и ныне там.	Раз ў калёсы запрагліся Лебедзь ды Шчупак і Рак; З усіх сіл вязці ўзяліся, Дый не зрушаць воз ніяк. Пад нябёсы Лебедзь рвецца, Шчупак цягне ў ваду; Распусціўшы клюшні, пхнецца Рак назад, як на бяду! ---------- Хто з іх вінен, хто не вінен, Не нам гэта раз'ясняць, Толькі ж і дасюль калёсы Як стаялі, так стаяць!

Верш Я. Купалы[1]	Пераклад верша на **французскую** мову M. Zakharkévitch[2]
А хто там ідзе? А хто там ідзе, а хто там ідзе У агромністай такой грамадзе? — Беларусы. А што яны нясуць на худых плячах, На руках у крыві, на нагах у лапцях? — Сваю крыўду. А куды ж нясуць гэту крыўду ўсю, А куды ж яны нясуць напаказ сваю? — На свет цэлы. А хто гэта іх, не адзін мільён, Крыўду несць наўчыў, разбудзіў іх сон? — Бяда, гора. А чаго ж, чаго захацелась ім, Пагарджаным век, ім, сляпым, глухім? — Людзьмі звацца.	**Qui vient par ici?** Qui vient par ici, qui vient par ici, Et qui est ce flot humain qui grossit? — Les Bélarussiens. Que portent-ils donc sur leur dos voûte? Que tiennent leurs bras tout ensanglantés? — L'injustice. Où la portent-ils, sur leur dos chargé, Qui doit la connaître et peut la juger? — Tout le monde. Mais tous ces millions, qui leur enseigna A voir l'injustice, et les réveilla? — La misère. Qui réclament-ils donc, aveugles, sourds, Asservis, esclaves depuis toujours? — Le nom d'hommes.

[1] По материалам сайта http://yankakupala.ru/khto-tam-idze (дата доступа 11.01.2018).
[2] Вадюшина, Д. С. Французский язык: учеб. пособие для 8-го кл. — Минск: Вышэйшая школа, 2016. С. 163.

Johan Wolfgang Goethe[1] Des Wanderers NachtLied	
Über allen Gipfel ist Ruh, in allen Wipfeln spürest du kaum einen Hauch; die Vögelein schweigen im Walde. Warte nut, balde ruhest du auch.	
Перевод[2] на **русский** язык М. Ю. Лермонтова	Перевод[3] на английский язык H. W. Longfellow
Горные вершины Спят во тьме ночной, Тихие долины Полны свежей мглой. Не пылит дорога, Не дрожат листы. Подожди немного — Отдохнешь и ты.	O'er all the hill-tops Is quiet now, In all the tree-tops Hearst thou Hardly a breath; The birds are asleep in the trees: Wait; soon like these Thou too shaft rest.

2 Оформите в виде таблицы расписание уроков на неделю.

3 Создайте таблицу спряжения глагола и оформите ее по образцу.

1. Have в английском языке

Person	Singular		Plural	
1-е лицо	I have		we	have
2-е лицо	you have		you	
3-е лицо	he	has	they	
	she			
	it			

2. Haben в немецком языке

Person	Singular		Plural
1-е лицо	ich habe		wir haben
2-е лицо	du hast		ihr habt
3-е лицо	er	hat	sie haben Sie haben
	sie		
	es		

[1] По материалам сайта http://www.poetarium.info/goethe/wn.htm (дата доступа 11.01.2018).

[2] Лермонтов, М. Ю. Сочинения в 2 т. Т. 1 / сост. и ком. И. С. Чистовой. М.: Правда, 1988 (с.197).

[3] По материалам сайта http://www.bartleby.com/356/524.html (дата доступа: 11.01.2018).

3. Avoir во французском языке

Personne	Singulier		Pluriel
1-е лицо	j'ai		nous avons
2-е лицо	tu as		vous avez
3-е лицо	il	a	ils ont
	elle		
	on		

4. Tener в испанском языке

Persona	Singular		Plural	
1-е лицо	yo tengo		nosotros (as) tenemos	
2-е лицо	tú tienes		vosotros (as) tenéis	
3-е лицо	él	tiene	ellos	tienen
	ella		ellas	
	Usted (Vd.)		Ustedes (Vds)	

4 Создайте таблицы, содержащие сведения из различных предметных областей. Оформите таблицу в соответствии с образцом или примените свое оформление.

1. Биология.

СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА КЛАССОВ ЧЛЕНИСТОНОГИХ			
	Ракообразные	**Паукообразные**	**Насекомые**
Среда обитания	В основном водная	Наземная	В основном наземная
Расчленение тела	Чаще головогрудь и брюшко	Головогрудь и брюшко	Голова, грудь и брюшко
Количество конечностей	Разное	Четыре пары	Три пары
Количество усиков	Две пары	Нет	Одна пара
Крылья	Нет	Нет	Две пары, реже — одна или нет
Глаза	Чаще сложные	Простые	Сложные и простые

2. История.

	Европа			США
	Англия	Франция	Германия	
Заработная плата в 1850 г. за равное рабочее время (в %)	100	64	75	240
Продолжительность рабочего времени в 1850 г. (в %)	100	117	111	96
Продолжительность рабочей недели в 1850 г. (в часах)	84			72

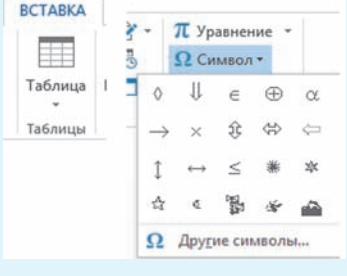
3*. География.

КЛИМАТИЧЕСКИЕ ПОЯСА

	Средние температуры (°С)		Осадки		
	Зима	Лето	Годовые, мм		Когда выпадают
			min	max	
Умеренный	0	+16	500	1000	В течение года
Субтропический	+8	+24	250	1000	Зимой
Экваториальный	+24	+24	500	2000	В течение года

§ 23. Вставка символов и формул

Пример 23.1. Выбор символа для вставки.



23.1. Вставка и размещение символов в текстовом документе

Современные компьютеры используют таблицу символов Unicode, содержащую 65 536 символов.

В Word поддерживается возможность вставки символов: команда **Символ** на вкладке **Вставка** (пример 23.1). Вставку символов выполняют тогда, когда необходимо добавить в текст сим-

вол, отсутствующий на клавиатуре, но имеющийся в таблице символов.

Если в выпадающем списке отсутствует нужный символ, то можно открыть окно **Символ**, выбрав команду Ω Другие символы... (пример 23.2). Для поиска символа в таблице можно воспользоваться выпадающими списками **Шрифт** и **Набор** (присутствует не для всех шрифтов). В примере 23.3 описана последовательность действий для вставки в текст символа π.

В таблице символов можно найти символы европейских и восточных языков, математические и нотные символы, символы денежных единиц и символы-картинки, которые можно вставить в текст (пример 23.4). К этим символам можно применять форматирование.

23.2. Создание и редактирование формул

Математические (физические, химические и др.) формулы могут содержать в себе достаточно сложные элементы: дроби, знаки корня, системы уравнений или неравенств. Для создания таких формул одних символов бывает недостаточно.

Для ввода формул в Word используют команду **Вставка → Уравнение → Вставить новое уравнение** (пример 23.5). После выполнения команды в тексте появится область для ввода формулы (уравнения):

Дополнительно откроется вкладка **Работа с уравнениями → Конструктор**,

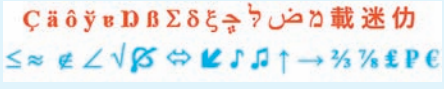
Пример 23.2. Выбор символа для вставки в окне **Символ**.

Пример 23.3. Вставка символа π.

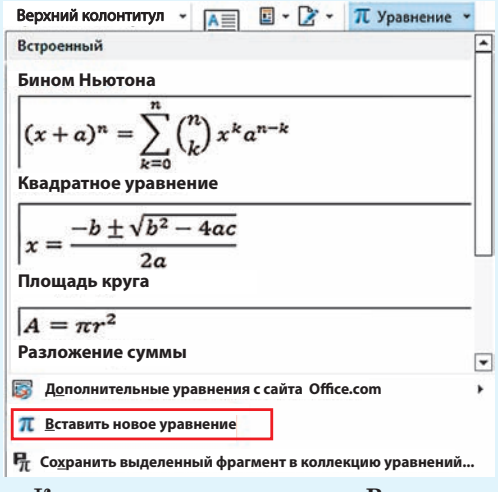
1. Выполнить команду **Вставка → → Символ → Другие символы.**
2. В поле **Шрифт** выбрать **Symbol.**
3. Найти символ π и нажать кнопку **Вставить.**

Число π (пи) — математическая константа, равная отношению длины окружности к ее диаметру.

Пример 23.4. Различные символы из таблицы символов:

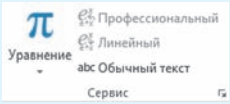


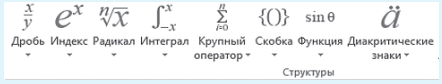
Пример 23.5. Вставка уравнения.



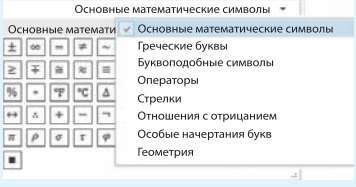
Команда может называться **Вставка → Формула → Вставить новую формулу.**

Пример 23.6. Группы вкладки **Работа с уравнениями → Конструктор.**





Пример 23.7. Категории символов.



Пример 23.8. Запись системы неравенств $\begin{cases} 2b-3 \geqslant 13, \\ 3b^2-1<1. \end{cases}$

1. **Вставка → Уравнение → Вставить новое уравнение.**

2. В группе **Структуры** открыть шаблон Скобка, в открывшемся списке выбрать **Отдельная скобка:** $\{\square$.

3. Открыть шаблон Матрица. Выбрать из выпадающего списка **Пустая матрица 2 × 1:** $\begin{matrix}\square\\\square\end{matrix}$. Получим $\{\begin{matrix}\square\\\square\end{matrix}$.

4. Ввести первое неравенство. Знак $\geqslant$ можно найти в группе **Символы.**

5. Ввести второе неравенство. Знак $<$ расположен на клавиатуре.

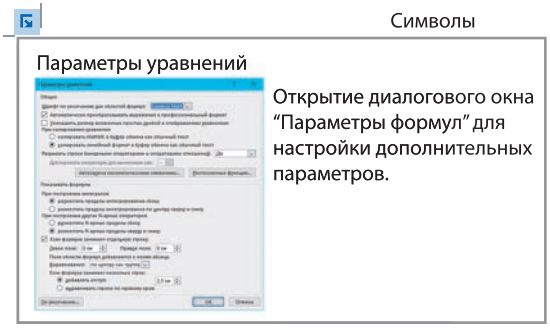
6. Для ввода b^2 использовать шаблон $\square^{\square}$.

7. Выровнять неравенства в системе по левому краю, используя контекстное меню объекта **Уравнение.**

на которой представлены команды и шаблоны для ввода различных элементов формулы.

На вкладке выделены три группы: **Сервис**, **Символы** и **Структуры** (пример 23.6).

Команды группы **Сервис** позволяют определить, как будет выглядеть формула, или вставить готовое уравнение из имеющихся шаблонов. Развернув группу с помощью маленькой кнопки со стрелочкой в нижнем правом углу ↘, получим доступ к общим настройкам параметров уравнений.



Большинство символов, отсутствующих на клавиатуре, но используемых для ввода формул, размещаются в группе **Символы.** Символы объединены по категориям: основные математические символы, греческие буквы, стрелки и др. (пример 23.7).

В группе **Структуры** находятся шаблоны для ввода дробей, индексов (верхних и нижних), корней, скобок и др.

В примере 23.8 показано, как с помощью инструмента **Уравнение** записать систему линейных неравенств:

$$\begin{cases} 2b-3 \geqslant 13, \\ 3b^2-1<1. \end{cases}$$

Для изменения формулы достаточно кликнуть по ней. Вновь станет доступна вкладка **Работа с уравнениями** → → **Конструктор.**

Контекстное меню объекта уравнение содержит команды, позволяющие редактировать и форматировать уже имеющуюся формулу.

Для вставки формул в текст документа, кроме редактора Word, можно воспользоваться другими текстовыми редакторами:

- LaTex[1] (имеет собственный язык верстки формул);
- MathType[2] (представляет собой небольшую программу, которая запускается вместе с Word на отдельной вкладке).

1. Как вставить в текст символ, отсутствующий на клавиатуре?
2. Как вставить формулу в текст документа?

Упражнения

1 Наберите тексты (или откройте тексты из файла). Используйте вставку символов, для тех символов, которые отсутствуют на клавиатуре. Следите за форматированием символов.

1. Если $a > 0$, то неравенство $|x| \leqslant a$ равносильно неравенству $-a \leqslant x \leqslant a$.
2. Пусть у $\triangle ABC$ и $\triangle A_1B_1C_1$ равны стороны AB и A_1B_1, BC и B_1C_1, но $AC > A_1C_1$. Докажем, что $\triangle ABC > \triangle A_1B_1C_1$.
3. Длина окружности вычисляется по формуле $l = 2\pi r$, а площадь круга — по формуле $S = \pi r^2$.
4. В физике популярна шкала Кельвина. В ней 0° С соответствует 273 К, а 100 °С — это 373 К.
5. При переходе 1 кг вещества из твердого состояния в жидкое поглощается количество теплоты, численно равное удельной теплоте плавления λ, и ровно столько же выделяется при его переходе из жидкого состояния в твердое.
6. Схема химической реакции: $Cu + O_2 \rightarrow CuO$.
7. При написании химических уравнений применяют также знак $\downarrow$, если вещество образует осадок, или знак $\uparrow$, если в результате реакции образуется газ. Например:

$$AgNO_3 + HCl = AgCl\downarrow + HNO_3$$
$$C + O_2 = CO_2\uparrow$$

[1] Можно бесплатно скачать на официальном сайте www.latex-project.org

[2] Ссылка для скачивания MathType: www.dessci.com. Имеется бесплатная 30-дневная версия.

2 Наберите тексты с формулами (или откройте тексты и вставьте формулы).

1. При решении уравнения $\sqrt{2x+3} = x$ возведем обе его части в квадрат. Получим: $\left(\sqrt{2x+3}\right)^2 = x^2$, или $2x + 3 = x^2$.

2. Площадь ромба можно вычислить по формуле $S = \frac{d_1 d_2}{2}$, где d_1, d_2 — длины диагоналей ромба.

3. Результаты опытов позволяют записать формулу для расчета сопротивления проводника: $R = \rho\frac{l}{S}$. Коэффициент ρ называют удельным сопротивлением вещества.

4. Параллельное соединение позволяет подключать к источнику независимо друг от друга различные приборы, несмотря на их рабочий ток. Если параллельных проводников только два, то: $\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} = \frac{R_1 + R_2}{R_1 R_2}$, что приводит к простому выражению: $R = \frac{R_1 R_2}{R_1 + R_2}$.

5. В промышленности водород получают, пропуская водяные пары над раскаленным углем: $C + H_2O \xrightarrow{t} CO + H_2$.

3 Создайте объявления (откройте файлы с текстом), используя символы шрифтов Webdings или Wingdings. Объявления удобно оформлять таблицей, делая некоторые границы ячеек невидимыми. Порассуждайте, всегда ли можно доверять таким объявлениям.

Туристическая фирма приглашает на отдых.

Проживание	Гостиничные номера	
	Номера люкс с удобной мебелью	
	Домики на берегу моря	
ПИТАНИЕ	Ресторан	
	Бар	
Досуг	Отдых на пляже	
	Катание на горных лыжах	
	Занятие в тренажерном зале	
Дополнительная информация	Вылет из Минска	
	Работает прокат автомобилей	
Наш телефон	**452-369-89**	

Планетарий приглашает любителей астрономии на лекцию

ЗОДИАКАЛЬНЫЕ СОЗВЕЗДИЯ

Вход свободный

Длительность – 1 час

тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89 тел. 927-56-89

§ 24. Графические объекты в текстовом документе

24.1. Вставка рисунков

Word позволяет вставлять в документ рисунки из различных источников:

- рисунки, хранящиеся на диске в графических файлах;
- изображения из Интернета;
- векторные рисунки, созданные с помощью фигур (графических примитивов);
- рисунки — графические копии экрана.

Вставить рисунок можно, пользуясь буфером обмена. Для этого в другой программе выделяем рисунок или его фрагмент и выполняем команду **Копировать**. Затем возвращаемся в Word и выполняем команду **Вставить**.

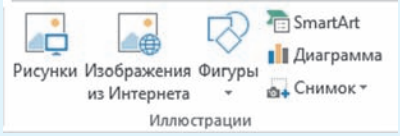
Вкладка **Вставка** содержит команды для размещения в текстовом документе различных видов иллюстраций (пример 24.1). Создание векторного изображения из автофигур выполняется аналогично созданию изображения в векторном графическом редакторе.

Для вставки рисунка из графического файла нужно выполнить команду **Рисунки**. Затем следует указать имя файла. Рисунок будет вставлен в позицию курсора.

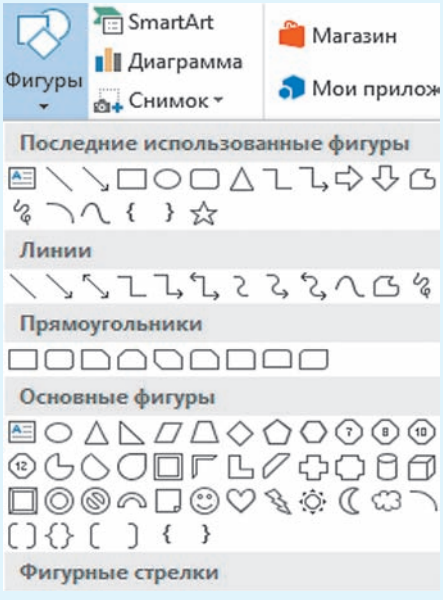
При выборе команды **Изображения из Интернета** в строке вводим запрос, отражающий содержимое рисунка (пример 24.2).

Для вставки копии экрана существуют несколько возможностей. Используя клавишу PrtScr[1] (копия всего экрана)

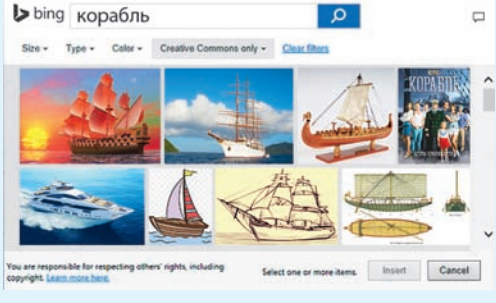
Пример 24.1. Команды для вставки иллюстраций на вкладке **Вставка**:



Коллекция графических примитивов **Фигуры**:

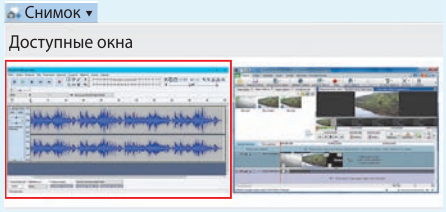


Пример 24.2. Поиск рисунка по команде **Изображения из Интернета**.

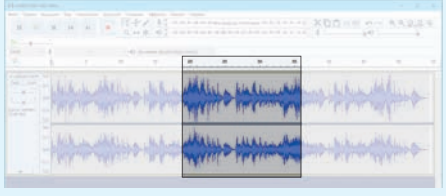


[1] На некоторых клавиатурах эта клавиша подписана Print Screen.

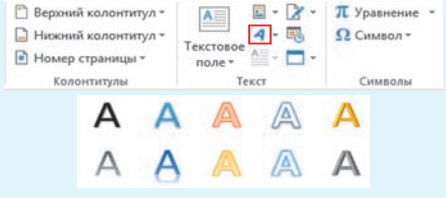
Пример 24.3. Использование команды **Снимок**.



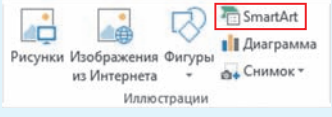
Пример 24.4. Использование команды **Вырезка экрана**. В документ будет вставлена выделенная часть экрана.

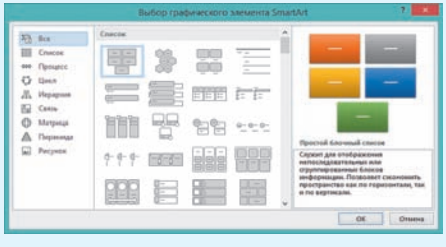


Пример 24.5. Вставка объекта WordArt и выбор стиля (из выпадающего списка).



Пример 24.6. Вставка объекта SmartArt и выбор стиля (из окна **Выбор графического элемента SmartArt**).





или комбинацию клавиш Alt + PrtScr (копия только активного окна), помещаем изображение в буфер обмена, а затем вставляем его в документ. Команда **Снимок** позволяет вставить в документ копию любого из открытых окон. Последнее из открытых окон отображается первым в списке команды **Снимок** (пример 24.3). Из него можно вырезать часть экрана (пример 24.4).

24.2. Вставка объектов WordArt и SmartArt

С объектами WordArt и SmartArt вы познакомились в 6-м классе, когда создавали презентации. Работа с этими объектами в редакторе Word происходит аналогично.

Для вставки объектов в текст выбирается соответствующая команда на вкладке **Вставка**. В примере 24.5 показано, как вставить объект WordArt, а в примере 24.6 — объект SmartArt.

При выделении объекта WordArt добавляется вкладка **Средства рисования** → **Формат**, на которой можно настроить внешний вид объекта. Для объекта WordArt можно изменить следующие параметры (пример 24.7):

- цвет, толщину и стиль линии контура вокруг символов текста;
- цвет или градиент для заливки;
- варианты тени;
- отражение и рельеф символов;
- подсветку вокруг символов;
- искривление текста.

При выделении объекта SmartArt (пример 24.8) добавляются две вкладки **Работа с рисунками SmartArt** → **Конструктор** и **Работа с рисунками**

SmartArt → Формат. Команды первой вкладки позволяют менять структуру объекта, а второй — его внешний вид. Подробнее о настройках объекта см. *Приложение 4* (с. 167).

24.3. Форматирование объектов

Мы рассмотрели вставку в текстовый документ различных объектов: WordArt, SmartArt, формул, рисунков. После вставки любого из этих объектов становится активной вкладка **Формат**. На ней содержатся команды, позволяющие выбрать параметры форматирования соответствующего объекта. Для разных объектов список этих команд различен. Однако есть команды, являющиеся общими для различных объектов. Эти команды объединены в две группы: **Упорядочение** и **Размер** (пример 24.9).

Команды группы **Упорядочение** позволяют управлять положением объекта и обтеканием текста.

Рисунок, вставленный в текстовый документ, можно обрезать с использованием инструмента **Обрезка**.

При разворачивании группы **Размер** (кнопка со стрелочкой ↘) получим отдельное окно **Макет**, позволяющее задавать параметры форматирования объектов. Окно имеет три вкладки: **Положение**, **Обтекание текстом** и **Размер**.

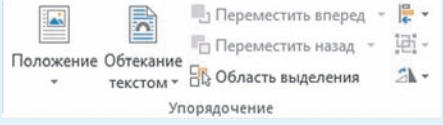
Вкладка **Размер** (пример 24.10) позволит изменить размер объекта: задать нужный размер в сантиметрах или в процентах относительно исходного размера. Изменять размер рисунка можно с помощью мыши.

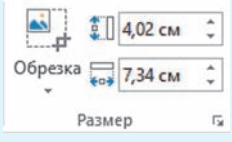
Пример 24.7. Пример текста WordArt:

С днем рождения!

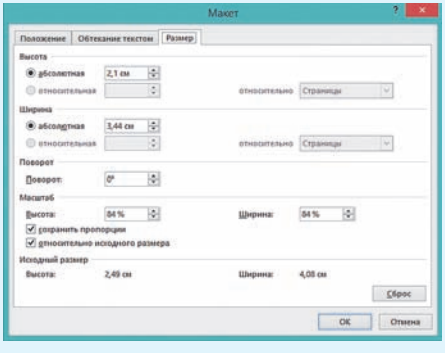
Пример 24.8. Пример рисунка SmartArt: организационная диаграмма.

Пример 24.9. Команды групп **Упорядочение** и **Размер** на вкладках **Форматирование:**

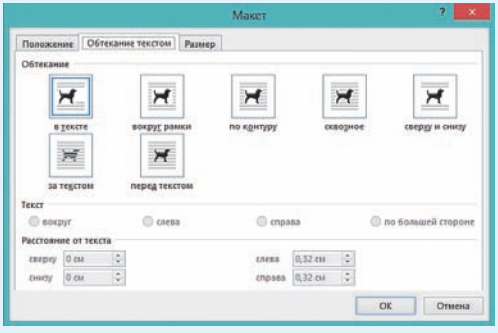




Пример 24.10. Вкладка **Размер:**



Пример 24.11. Вкладка **Обтекание текстом:**



Пример 24.12. Обтекание рисунка **Сквозное**, текст — вокруг:

А. С. Грин. Алые паруса[1]

— Не знаю, сколько пройдет лет, — только в Каперне расцветет одна сказка, памятная надолго. Ты будешь большой  Ассоль. Однажды утром в морской дали под солнцем сверкнет алый парус. Сияющая громада алых парусов белого корабля двинется, рассекая волны, прямо к тебе.

Пример 24.13. Обтекание рисунка **Вокруг рамки**, текст — слева:

В. Быкаў. Альпійская балада[2]

Сям-там сінелі лапікі буйных духмяных незабудак, калыхаліся на ветры званочкі, ад  густога водару жоўтай азаліі п'янела ў галаве. Мясцінамі ў кветкавых чашчобах траплялiся жарсцявыя плешыны, тырчалі з травы шэрыя камяні, каля іх заўжды было многа калючага шчэбню, ён шкодзіў ступням.

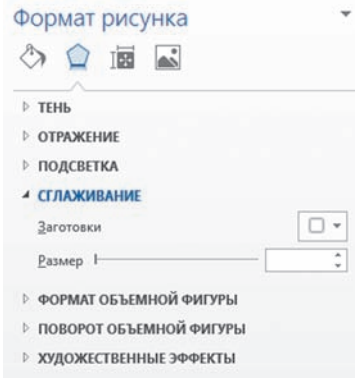
На вкладке **Обтекание Текстом** (пример 24.11) можно установить обтекание объекта текстом (в тексте, вокруг рамки, сквозное и др.) и выравнивание относительно текста. Положение **в тексте** размещает объект как символ текста.

Вкладка **Положение** позволит определить положение объекта в документе относительно страницы, абзаца или колонки.

Примеры 24.12 и 24.13 демонстрируют один из вариантов обтекания текстом.

Обтекание текста можно определить с помощью значка  , который появляется рядом с объектом при его выделении.

Для любого объекта можно выбрать команду **Формат** из контекстного меню, которая открывает дополнительную панель. На этой панели собраны все параметры форматирования соответствующего объекта. Панель дублирует соответствующие команды вкладки **Формат.** Внешний вид панели может различаться для различных объектов. Панель **Формат рисунка** выглядит следующим образом:



[1] Грин, А. С. Алые паруса. Минск: Наука и техника, 1979, 384 с.
[2] Быкаў, В. Збор твораў. У 4 т. Т. 1. Аповесці. Мінск: Мастацкая літаратура, 1980, 432 с.

1. Как можно вставить рисунок в текстовый документ?
2. Какие способы вставки копии экрана в текстовый документ вам известны?
3. Какие параметры объекта WordArt можно изменять?
4. Какие параметры объекта SmartArt можно изменять?
5. Как изменить размеры рисунка?
6. Какие способы обтекания текстом вы знаете?
7. Как открыть панель **Формат рисунка**?

Упражнения

1 Используя поиск рисунков (или заранее заготовленные рисунки), создайте поздравления. Для оформления надписей используйте объект WordArt.

С Днем труда

С Днем Победы

2 Создайте с помощью объектов WordArt и SmartArt (тип **Рисунок**) следующую схему. Изображения получите с экранной копии.

3 Проиллюстрируйте текст, указанный учителем. Для иллюстраций можно использовать рисунки, хранящиеся на компьютере, изображения из Интернета или самостоятельно нарисовать рисунок в графическом редакторе и вставить его из файла или используя буфер обмена.

4 Откройте файл с текстом. Проиллюстрируйте текст, используя копии экрана или готовые рисунки.

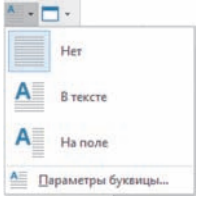
Как вставить буквицу в документе Word[1]

Хотите добавить изюминку в документы Word? Буквица и есть такая изюминка, которая позволяет отображать первую букву абзаца большим шрифтом.

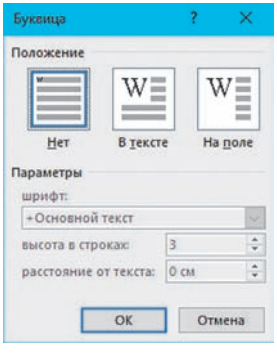
1. Поместите курсор в начало абзаца, в который хотите вставить буквицу.

2. Откройте вкладку **Вставка** и в группе **Текст** нажмите **Буквица**.

3. Выберите тип буквицы: **В тексте** или **На поле**.



4. В разделе **Параметры буквицы** выберите шрифт буквицы.



5. Задайте высоту в строках и расстояние от текста в соответствующих полях в окне **Буквица**.

6. Нажмите **ОК**, чтобы вставить буквицу.

Биткоин (англ. *Bitcoin*, от *bit* — «бит» и *coin* — «монета») — платежная система, использующая одноименную единицу для учета операций и одноименный протокол передачи данных.

[1] По материалам сайта https://hi-news.ru/gadgets (дата доступа 16.01.2018).

Гаджеты[1]

Гаджеты — технические приспособления для совершенно разных целей — являются лучшими друзьями человека. По статистике на каждого живущего на Земле приходится минимум три технических устройства, т. е. полезных гаджета. Будучи совершенно разными и предназначенными для разных целей (часы и телефоны, док-станции и микрофоны, проекторы и миниатюрные роботы), гаджеты существенно облегчают жизнь человека.

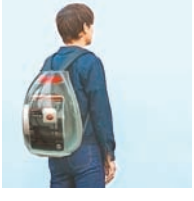
Квадрокоптеры научили летать стаями без GPS

В Интернете много роликов, где стаи дронов красиво и слаженно летят, попутно выполняя различные трюки. Ну и что? Они ведь опираются на данные, полученные по GPS. Но что делать, когда никакого GPS использовать не получается? Инженеры университета Пенсильвании решили научить дроны обходиться без них.

Беспроводное зарядное устройство WattUp работает на расстоянии до 4 м

Инженеры уже давно пытаются создать зарядное устройство, которое позволит людям раз и навсегда избавиться от проводов и постоянной необходимости находиться рядом с розеткой. Учитывая, сколько в нашей жизни всевозможных гаджетов и электронных устройств, нам приходится регулярно подзаряжать их, чтобы оставаться на связи, в курсе последних новостей и т. д. Новое зарядное устройство WattUp, разработанное в рамках стартапа Energous, позволяет подзаряжать по воздуху сразу несколько гаджетов на расстоянии более 4 м.

Представлен самый компактный 3D-принтер размером с рюкзак



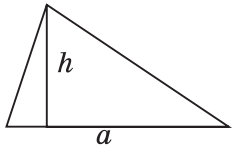
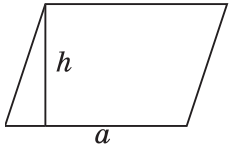
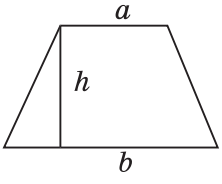
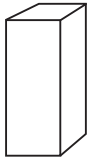
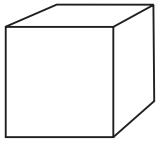
Несмотря на то что 3D-принтеры все активнее входят в нашу жизнь, они все же остаются достаточно громоздкими устройствами. Однако китайские инженеры из корпорации MakeX представили самый компактный на данный момент 3D-принтер. Интересно то, что этот принтер встроен в обычный рюкзак.

[1] По материалам сайта https://hi-news.ru/gadgets (дата доступа: 23.01.2018).

5 Вставьте таблицу из трех строк и двух столбцов. В одной ячейке таблицы создайте свою визитную карточку. Укажите фамилию, имя, школу и класс. Вставьте свою фотографию. Скопируйте визитку во все ячейки таблицы.

6 Вставьте таблицу из пяти строк и двух столбцов. В одной ячейке таблицы создайте бейдж дежурного. Скопируйте бейдж во все ячейки таблицы.

7 Создайте следующую таблицу. Формулы создаются с помощью вставки уравнений, рисунки — с помощью вставки фигур.

Формулы для вычисления площади			
Треугольник	$S = \frac{ah}{2}$	a — основание треугольника h — высота треугольника	
Параллелограмм	$S = ah$	a — основание параллелограмма h — высота параллелограмма	
Трапеция	$S = \frac{a+b}{2}h$	a, b — основания трапеции h — высота трапеции	
Формулы для вычисления объема			
Прямоугольный параллелепипед	$V = abc$	a, b, c — длина, ширина, высота параллелепипеда	
Куб	$V = a^3$	a — длина ребра куба	

8* Оформите с помощью текстового редактора Word решение задачи по геометрии, физике или химии. Используйте таблицы, формулы, рисунки.

§ 25. Использование стилей

25.1. Понятие стиля

При работе с большими текстовыми документами задание свойств символов и абзацев является довольно трудоемким процессом. Использование стилей позволяет значительно ускорить форматирование текста.

Под **стилем** понимают набор параметров форматирования текста. Стиль имеет свое имя.

Помимо чисто оформительской задачи, стили позволяют решить также задачу структурирования текста. Для этого каждый из стилей сопоставляют с разделом, названием заголовка, текстом основной части документа и др. Выделяют следующие виды стилей:

- стиль абзаца;
- стиль символа;
- стиль списка.

Форматирование с использованием стилей (стилевое форматирование) имеет ряд преимуществ перед ручным:

1. Позволяет экономить время. Применить стиль как набор параметров форматирования значительно быстрее, чем последовательно применять отдельные параметры.

2. Способствует единообразному оформлению текстового документа.

3. Позволяет быстро изменить вид отдельных элементов во всем документе. В этом случае достаточно внести изменения в стиль, и оформление автоматически применится ко всему документу.

Текстовый процессор Word позволяет пользователю создавать свои стили или использовать уже созданные. Для работы со стилями на вкладке **Главная** имеется группа **Стили** (пример 25.1).

Обычно новые документы создаются на основе шаблона normal.dotx, который содержит определенные стили.

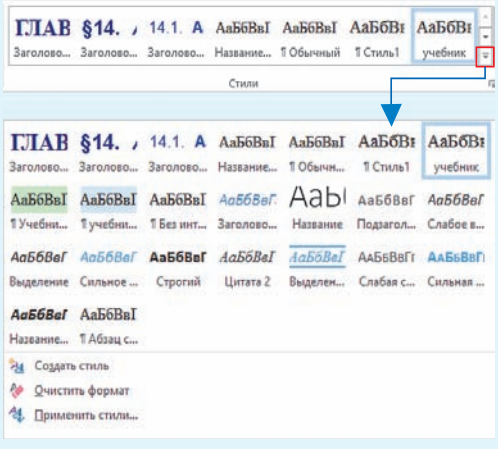
Стили текста (абзаца или символа). Как правило (или часто), используют стили **Обычный** и **Основной текст**, а также их модификации. На их основе определяются стили абзацев и символов.

Стили заголовков (особый стиль абзаца). Применение заголовочных стилей позволяет автоматизировать создание оглавления. Заголовок рассматривается как абзац.

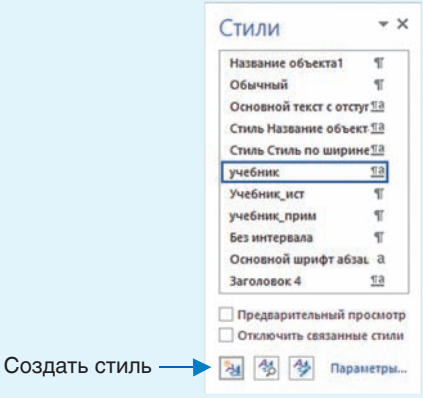
Стили списков. Позволяют оформлять маркированные и нумерованные списки.

Каждый тип стиля определяет только присущие ему параметры. Соответственно стили разных типов могут накладываться друг на друга. Например, стили списков управляют видом маркеров, структурой списка и величиной отступов пунктов, но не размером шрифта. Таким образом, текст в списке управляется (как минимум) двумя стилями — стилем абзаца и стилем списка.

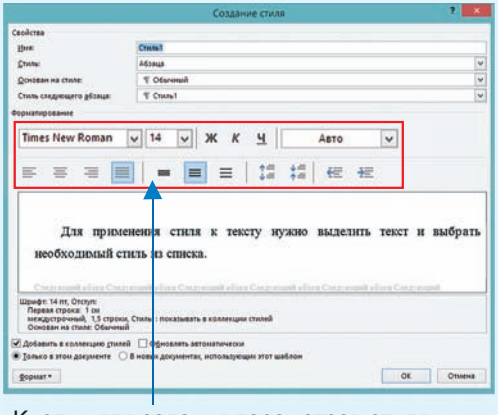
Пример 25.1. Группа **Стили** вкладки **Главная**:



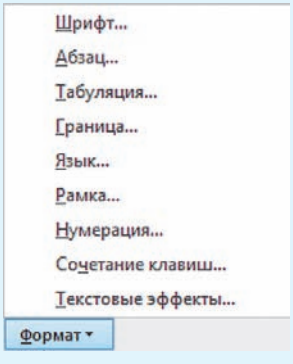
Пример 25.2. Панель **Стили**.



Пример 25.3. Панель **Стили**.



Пример 25.4. Меню кнопки **Формат**.



Список имеющихся стилей отображается также на панели **Стили** (пример 25.2), которая разворачивается с помощью кнопки со стрелочкой . Для применения стиля к тексту нужно выделить текст и выбрать необходимый стиль из списка.

Для создания стиля используют кнопку **Создать стиль** в окне **Стили** (или команду **Создать стиль** в развернутом списке группы **Стили**). В открывшемся окне **Создание стиля** (пример 25.3) вводят необходимые параметры стиля.

В поле **Имя** задается название стиля (по умолчанию **Стиль с номером**).

В поле **Стиль** выбирается абзац, знак или таблица в зависимости от того, для какого объекта определяем стиль.

В поле **Основан на стиле** выбирают стиль, наиболее близкий по оформлению и назначению к создаваемому. Так, например, стили основного текста документа основывают на стиле **Обычный**, а стили для оформления названий глав, параграфов и т. д. основывают на стилях **Заголовок**.

В качестве **Стиля следующего абзаца** можно выбрать любой из уже имеющихся стилей. Как правило, для стилей, основанных на стиле **Обычный**, выбирают имя создаваемого стиля. Тогда все абзацы документа будут оформлены одним стилем (параметры стиля для нового абзаца переносятся из предыдущего). Для стилей, основанных на стиле **Заголовок**, в качестве стиля следующего абзаца определяют стиль заголовка другого уровня или

стиль текста, основанный на стиле **Обычный**.

Параметры форматирования символов и абзацев устанавливаются с помощью соответствующих кнопок. Можно также воспользоваться кнопкой **Формат** (пример 25.4).

Требования, предъявляемые к оформлению основного текста рукописи учебного пособия, включают следующие параметры текста: Шрифт Times New Roman, размер 14. Выравнивание абзаца по ширине, абзацный отступ — 1 см, междустрочный интервал — 1,5 строки. Создание стиля с именем **«Учебник»**, отвечающего указанным требованиям, описано в примере 25.5.

Созданный стиль можно удалить, изменить или обновить на основе выделенного фрагмента. Для этого используют команды контекстного меню для выбранного стиля (пример 25.6).

25.2. Стилевое оформление заголовков

Любой, даже самый простейший, документ состоит из различных разделов. Под разделом понимают часть текста, несущую определенный функциональный смысл. Понимание структуры документа дает возможность грамотно его оформить и без труда переформатировать в случае необходимости.

Например, учебное пособие состоит из глав, глава содержит параграфы, в параграфах могут быть пункты и подпункты (пример 25.7).

Так как одни разделы являются частями других (глава состоит из параграфов, параграфы состоят из пунктов), то разделы различают по уровням.

Пример 25.5. Создание стиля **«Учебник»**.

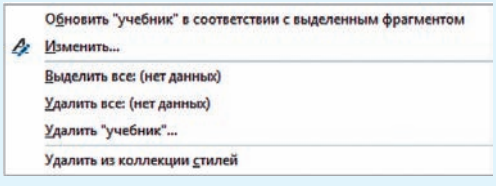
1. В окне **Стили** нажать кнопку **Создать стиль** и определить следующие параметры стиля: **Имя** — **Учебник**, **Стиль** абзаца, **Основан на стиле** — **Обычный**, **Стиль следующего абзаца** — **Учебник**.

2. Определить шрифт (Times New Roman), размер шрифта (14), выравнивание (по ширине) и междустрочный интервал (1,5 строки).

3. Для определения отступа первой строки нажать кнопку **Формат** и выбрать **Абзац...** В открывшемся окне **Абзац** задать размер отступа первой строки (1 см).

4. Закончить создание стиля, нажав кнопку OK. Просмотреть список стилей, найти созданный стиль. Стиль можно использовать для оформления документов.

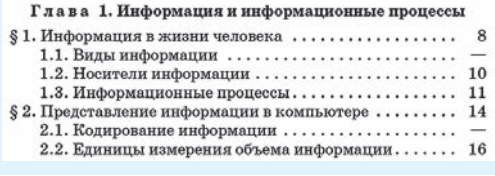
Пример 25.6. Контекстное меню стиля **«Учебник»**.



Инструмент **Формат по образцу** () позволяет быстро скопировать стиль с одного объекта текстового документа на другой.

Инструмент **Удалить все форматирование** () применяет к выделенному тесту стиль **Обычный**.

Пример 25.7. Структурные элементы учебного пособия «Информатика» для 7-го класса.

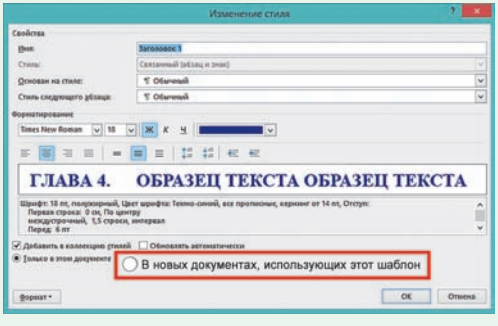


Пример 25.8. Параметры стиля **Заголовок 1** (структуру стиля можно просмотреть при наведении указателя мыши на его название в панели **Стили**).

Заголовок 1:
Шрифт
ШРИФТ 18 пт, полужирный, Цвет шрифта: Темно-синий
ВИДОИЗМЕНЕНИЕ все прописные
Интервал между знаками: кернинг от 14 пт
Абзац
ОТСТУПЫ
Первая строка: 0 см
ВЫРАВНИВАНИЕ По центру
ИНТЕРВАЛ:
междустрочный, 1,5 строки
Перед: 6 пт
после: 6 пт
РАЗРЫВЫ СТРОК И СТРАНИЦ: Не отрывать от следующего, Без переноса
Уровень структуры: Уровень 1
ТАБУЛЯЦИЯ
Поз.табуляции: 0 см, Выровнять по позиции табуляции
Список
Список: многоуровневый
Уровень: 1
Стиль нумерации: 1, 2, 3, ...
Начать с: 4
Выравнивание: слева
Выровнять по: 0 см
Табуляция после: 0 см
Отступ: 0 см
Стиль
Стиль Связанный, : показывать в коллекции стилей
Основан на стиле: Обычный
Следующий стиль: Обычный

Пример 25.9. Оформление **Заголовков** (изменение параметров стиля).

Если созданный стиль нужно использовать в других документах, то его сохраняют в шаблоне normal.dotx. Для этого выбирают пункт **В новых документах, использующих этот шаблон** в окне **Создание** или **Изменение стиля**. При закрытии документа будет предложено сохранить шаблон.



Раздел, входящий в состав другого, имеет уровень ниже.

Выделение структурных элементов текста обеспечивает структурирование документа, что облегчает его восприятие. Названия разделов оформляют заголовками соответствующего уровня: заголовок 1-го уровня, заголовок 2-го уровня и т. д. Одинаковые структурные элементы (например, название глав, параграфов, пунктов) должны оформляться одинаково — одним стилем.

Для заголовков применяют стили **Заголовок 1**, **Заголовок 2** (или стили, основанные на стилях **Заголовок**) и т. д. Номер в названии стиля заголовка соответствует его уровню. Например, для оформления заголовков глав в учебном пособии применяют стиль **Заголовок 1**, для заголовков параграфов — **Заголовок 2** (пример 25.8) и т. д.

На основе имеющихся стилей заголовков создают свои стили (пример 25.9). Для оформления заголовков, как правило, применяют шрифт большего размера, чем основной текст документа, и полужирного начертания (это позволяет сократить время поиска заголовков в тексте). Выравнивают заголовки обычно по центру (возможно также по левому или правому краям). Заголовки более высокого уровня оформляются более весомо, чем заголовки менее высокого уровня (крупнее размер, более жирное начертание и т. д.).

25.3. Генерация оглавления

Оглавление является обязательным элементом документа, в котором больше 10 страниц. Оно упрощает работу с документом.

Оглавление — указатель заголовков в документе, отражающий его структуру и ускоряющий поиск частей произведения по номерам страниц.

Перед тем как создавать оглавление, все заголовки глав, параграфов должны быть оформлены соответствующими стилями. В оглавление помещаются абзацы, оформленные стилем **Заголовок**, с указанием номера страниц, с которых они были взяты.

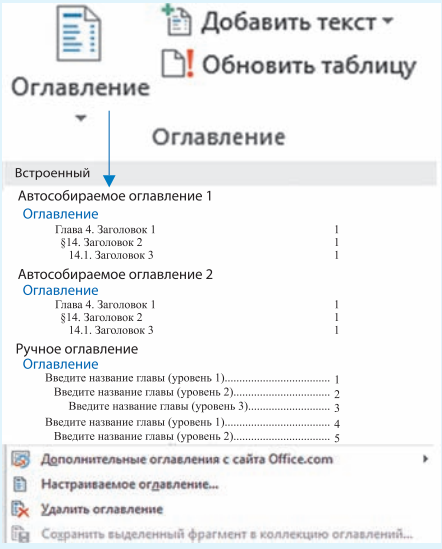
Оглавление создается с помощью команд группы **Оглавление** вкладки **Ссылки** (пример 25.10). В выпадающем списке команды **Оглавление** можно выбрать вид оглавления.

Автособираемое оглавление можно выбирать, если заголовки структурных элементов были оформлены стилями **Заголовок 1**, **Заголовок 2** и **Заголовок 3**. Если применялись другие стили для заголовков (**Заголовок 4—9**) или нужно изменить внешний вид оглавления, используют команду **Настраиваемое оглавление**, которая открывает окно **Оглавление** (пример 25.11).

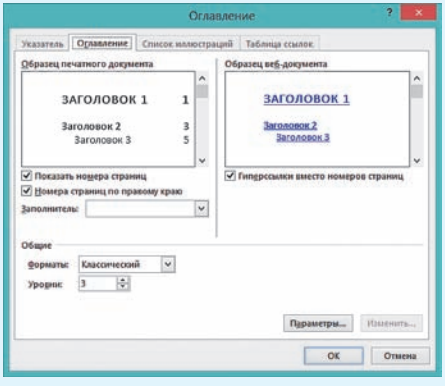
Внешний вид оглавления можно выбрать из списка **Форматы** (пример 25.12). С помощью кнопки **Параметры** (пример 25.13 на с. 146) задают те стили, на основе которых будет построено оглавление. Количество уровней заголовков, которые будут включены в оглавление, выбирают в выпадающем списке **Уровни**.

По умолчанию оглавление вставляется с номерами страниц, расположенными по правому краю. Птички в соответствующих полях позволяют располагать

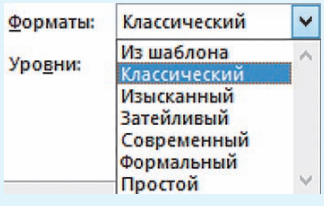
Пример 25.10. Группа **Оглавление** вкладки **Ссылки**.



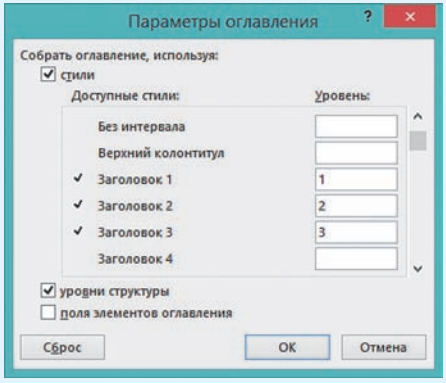
Пример 25.11. Окно **Оглавление**.



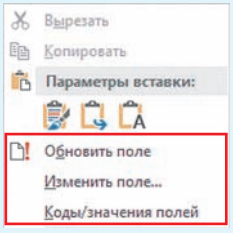
Пример 25.12. Форматы оглавления.



Пример 25.13. Параметры оглавления.



Пример 25.14. Контекстное меню оглавления.



номер страницы рядом с текстом или вообще убрать номера страниц.

В качестве заполнителя между заголовком и номером страницы обычно используют точки, но в выпадающем списке **Заполнитель** можно выбрать тире или знак подчеркивания. По умолчанию заполнитель отсутствует.

Оглавление обычно вставляют на отдельную страницу в начале или в конце документа. Для вставки новой страницы можно воспользоваться командой **Вставка → Пустая страница (Разрыв страницы)** или комбинацией клавиш Ctrl + Enter.

Если в документ внесли изменения, то оглавление нужно обновить. Управляют изменениями из контекстного меню оглавления (пример 25.14) или с помощью команды **Обновить таблицу** группы **Оглавление** на вкладке **Ссылки.**

1. Что понимают под стилем?
2. Для чего применяют стили?
3. Как создать и изменить стиль?
4. В чем особенность использования заголовочных стилей?
5. Как сгенерировать оглавление документа?

Упражнения

1 Откройте документ. Выполните следующие задания.

1. Очистите текст от примененных ранее стилей.
2. Создайте стиль **Доклад** (к названию **Доклад** приписать свою фамилию) со следующими параметрами: основан на стиле **Обычный**, стиль абзаца; шрифт Times New Roman, размер шрифта — 14; межстрочный интервал — полуторный, выравнивание абзаца — по ширине, отступ первой строки — 1 см; стиль следующего абзаца — **Доклад.**
3. Примените созданный стиль к основному тексту документа.
4. Оформите в виде нумерованного списка общий сценарий поведения пользователей в социальных сетях (пункт **Как работает социальная сеть**) и список источников.

5. Оформите в виде маркированного списка последовательность подвидов социальных сетей.
6. Отформатируйте таблицу.
7. Сохраните документ под новым именем.

2 Продолжите работу с документом из задания 1.

1. Создайте стиль заголовков первого уровня. Стиль **Заголовок 1 Доклад** с параметрами:
- основан на стиле **Заголовок 1**, стиль абзаца;
- шрифт — Comic Sans MS, размер шрифта — 22, начертание — полужирное, цвет — темно-синий;
- межстрочный интервал — полуторный, выравнивание абзаца — по центру, отступ первой строки — нет, интервал перед абзацем — 18, после — 12;
- стиль следующего абзаца — **Доклад**.

2. Примените стиль для заголовков: «Введение», «Социальные сети» и «Заключение».
3. Создайте стиль заголовков второго уровня. **Заголовок 2 Доклад** с параметрами:
- основан на стиле **Заголовок 2**, стиль абзаца;
- шрифт — Arial, размер шрифта — 18, начертание — полужирное, курсивное, цвет — синий;
- межстрочный интервал — полуторный, выравнивание абзаца — по центру, отступ первой строки — нет, интервал перед абзацем — 12, после — 6;
- стиль следующего абзаца — **Доклад**.

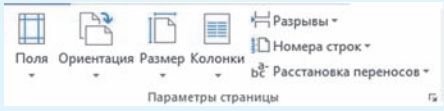
4. Примените стиль для заголовков: «Что такое социальные сети?», «Как работает социальная сеть?», «Подвиды социальных сетей», «Опасности социальных сетей», «Крупнейшие социальные сети», «Список источников».
5. Вставьте нумерацию страниц.
6. Сгенерируйте оглавление.
7. Сохраните документ.

3 Откройте документ, сохраненный в предыдущем задании.

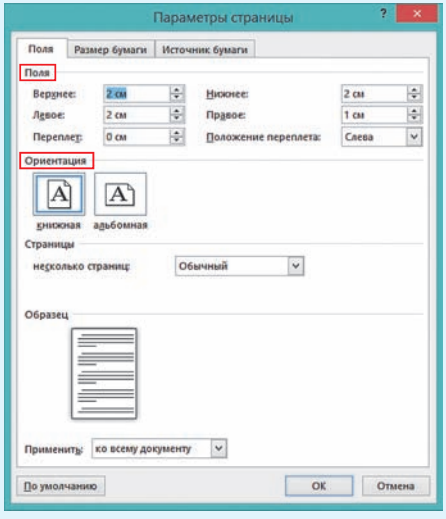
1. Измените в стиле **Доклад** размер шрифта с 14 на 15.
2. Измените в Стиле **Заголовок 1 Доклад** цвет шрифта с темно-синего на любой другой, а размер шрифта с 22 на 24.
3. Внесите какое-либо изменение в форматирование стиля **Заголовок 2 Доклад**.
4. Обновите оглавление.
5. Сохраните документ под новым именем.

§ 26. Форматирование страницы

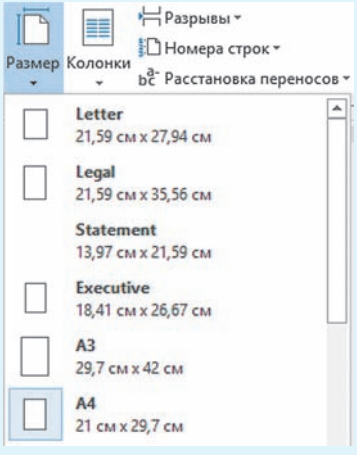
Пример 26.1. Команды группы **Параметры страницы.**



Пример 26.2. Окно **Параметры страницы.**



Пример 26.3. Команды кнопки **Размер.**



26.1. Параметры страницы

Подготовка текстового документа к печати начинается с форматирования страниц документа. Установить параметры форматирования страниц можно с помощью команд группы **Параметры страницы** на вкладке **Разметка страницы** (пример 26.1). Кнопка со стрелочкой ↘ разворачивает одноименное окно (пример 26.2).

При оформлении текстового документа, предназначенного для печати на бумаге, важным параметром документа является размер печатного листа (вкладка **Размер бумаги** или выпадающий список у кнопки **Размер** — пример 26.3). В большинстве случаев используется бумага стандартных размеров: А4 — 210 × 297 мм.

Для бумаги стандартного размера, кроме размеров, определяют и параметр **Ориентация** (кнопки **Поля** или **Ориентация** на вкладке **Разметка страницы**, пример 26.4). Различают книжную (вертикальную) ориентацию, при которой высота листа больше его ширины, и альбомную (горизонтальную), при которой ширина листа больше его высоты.

Еще одним параметром страницы документа, предназначенного для вывода на печать, является понятие **поля** (вкладка или кнопка **Поля**). Поле текстового документа — это расстояние от края листа до границы расположения текста на странице. Задают верхнее, нижнее, левое и правое поля. При выборе полей следует учитывать возможности принтера, требования стандартов (пример 26.5) и назначение документа.

Поля слева оставляют для переплета. Поля сверху и снизу обычно используют для колонтитулов и нумерации страниц.

Для оформления внешнего вида страниц можно использовать различные границы (пример 26.6). Окно **Границы и заливка** можно открыть с помощью соответствующей команды из выпадающего списка кнопки **Границы** (⊞) на вкладке **Главная**. Для границ можно использовать линии различного типа, цвета и ширины (вкладка **Страница**). Кроме того, в качестве границы можно выбрать различные изображения. В выпадающем списке **Рисунок** можно выбрать картинки, орнаменты и другие шаблоны границ страницы.

26.2. Колонтитулы

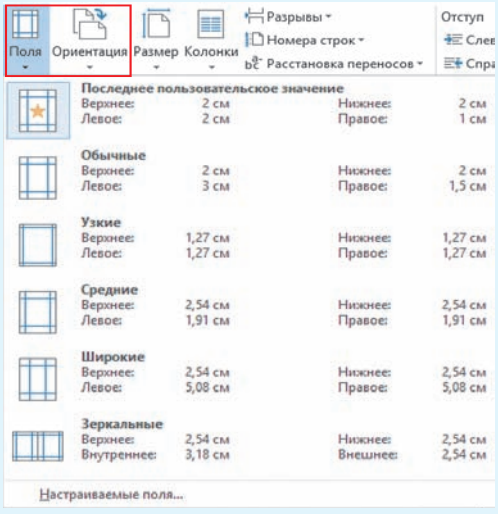
Отдельными параметрами текстового документа являются верхний и нижний колонтитулы.

Колонтитул — текст или изображение, которые размещаются на краю каждой страницы документа и повторяются на всех страницах.

Колонтитул обычно размещают на всех страницах документа, кроме титульных. Традиционно применяют верхний и нижний колонтитулы. В нижнем колонтитуле часто ставят нумерацию страниц документа, а в верхнем выводят название документа и (или) фамилию автора.

Внести или изменить информацию в разделе колонтитулов можно с помощью команд группы **Колонтитулы**

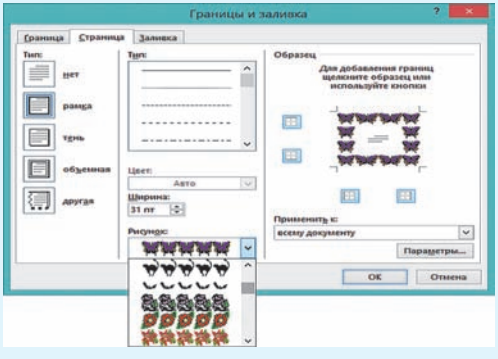
Пример 26.4. Команды кнопки **Поля**.



Пример 26.5. Стандарты оформления полей.

Для представления работ на XXII Республиканский конкурс работ исследовательского характера (конференция) учащихся[1] определены требования: лист формата А4; размер левого поля 30 мм, правого — 10 мм, верхнего и нижнего — 20 мм. Обычно такие же поля устанавливают для рефератов и других учебных работ.

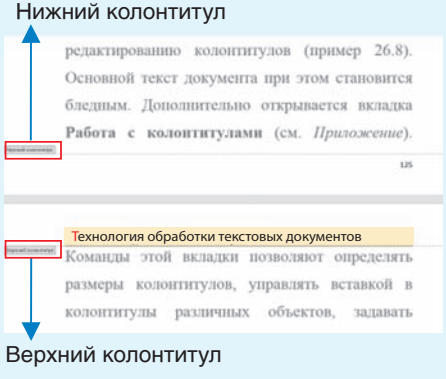
Пример 26.6. Окно **Границы и заливка**.



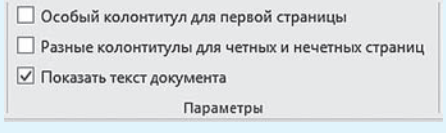
[1] http://www.uni.bsu.by/arrangements/conf/index.html (дата доступа: 16.01.2018).

Пример 26.7. Группа **Колонтитулы** вкладки **Вставка**.

Пример 26.8. Режим работы с колонтитулами.



Пример 26.9. Группа **Параметры** вкладки **Работа с колонтитулами**.



вкладки **Вставка** (пример 26.7). Команды **Верхний (Нижний) колонтитул** содержат несколько шаблонов оформления колонтитулов.

Двойной клик мыши в верхнем или нижнем поле страницы позволяет также перейти к редактированию колонтитулов (пример 26.8). Основной текст документа при этом становится бледным. Дополнительно открывается вкладка **Работа с колонтитулами** (см. *Приложение 4*, с. 167). Команды этой вкладки позволяют определять размеры колонтитулов, управлять вставкой в колонтитулы различных объектов, задавать параметры (пример 26.9). Параметр **Особый колонтитул** для первой страницы позволяет удалить колонтитул с первой страницы документа. Это важно, если первая страница является титульной.

Параметр **Разные колонтитулы** для четных и нечетных страниц позволяет вносить в колонтитулы различный текст (например, на левую страницу — заголовок, на правую — фамилию автора). Кнопка **Закрыть окно колонтитулов** возвращает в основной текст документа.

Для вставки номера страницы необходимо выполнить команду **Номер страницы** (пример 26.10). Для каждого местоположения номера страницы есть возможности дополнительно выбрать, где будет размещаться номер в колонтитуле и как он будет выглядеть (пример 26.11).

Команда **Формат номеров страниц** открывает одноименное окно. Здесь

можно выбрать формат номера: цифры или латинские буквы. К номеру страницы можно добавить номер главы. Здесь же можно определить, следует ли продолжать нумерацию страниц или начать ее заново (например, для нового раздела). Если нумерация начинается заново, то можно указать, с какого номера.

26.3. Подготовка документа к печати

Перед тем как отправить документ на печать, рекомендуется выполнить его предварительный просмотр.

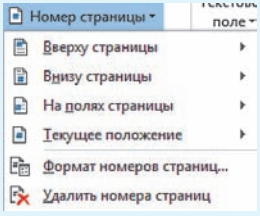
Предварительный просмотр документа позволяет пользователю увидеть, как будет выглядеть каждая страница документа при печати. После предварительного просмотра при необходимости можно внести изменения в оформление документа. Например, убрать лишние пустые страницы или изменить расположение абзацев.

Для вывода документа на печать необходимо выполнить команду **Файл** → → **Печать**. На панели **Печать** можно:

- указать количество копий документа;
- задать номера страниц, которые нужно вывести на печать;
- выбрать ориентацию и размер листа;
- выбрать количество страниц для печати в уменьшенном виде на одном листе;
- изменить размеры полей;
- выбрать принтер и настроить его свойства.

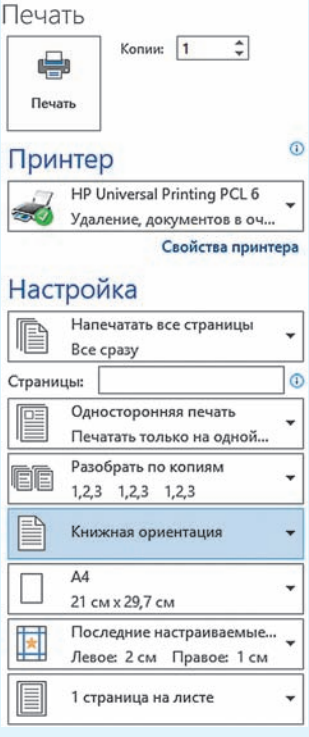
(Рассмотрите пример 26.12.)

Пример 26.10. Вставка номера страницы.



Пример 26.11. Номер внизу страницы.

Пример 26.12. Настройки печати.



1. Какие параметры страницы вы можете назвать?
2. Что такое поля страницы?
3. Какую ориентацию можно задать для страницы?
4. Что такое колонтитул?
5. Как вставить границу на страницу?
6. Как вставить номер страницы?

Упражнения

1 Установите для листа бумаги А4 альбомную ориентацию, все поля — по 2 см. Наберите текст. Подберите форматирование символов и абзацев так, чтобы текст занял весь лист. Выполните предварительный просмотр документа. Распечатайте.

Внимание! Внимание! Внимание!

17 мая в школе проводился СБОР МАКУЛАТУРЫ.

РЕЗУЛЬТАТЫ

1-е место — 8 Б класс — 300 кг

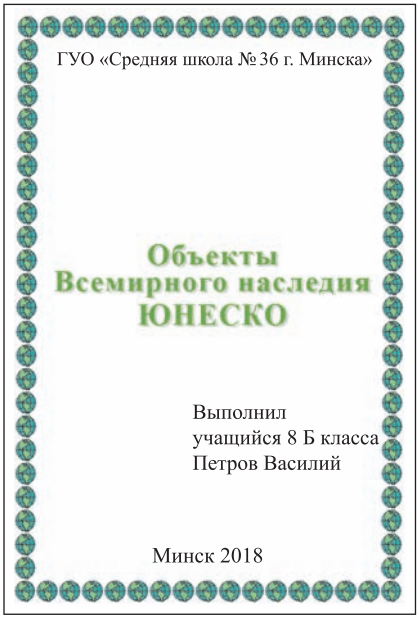
2-е место — 6 А класс — 250 кг

3-е место — 7 Г класс — 150 кг

Поздравляем победителей, которые получат *сладкие призы*.

Совет школы

2 Оформите титульный лист к реферату по географии по образцу на рисунке справа и распечатайте его на принтере. Для определения положения элементов титульного листа используйте линейку и команды вкладки **Разметка страницы**.



3 Откройте документ, созданный в задании 1 из предыдущего параграфа.

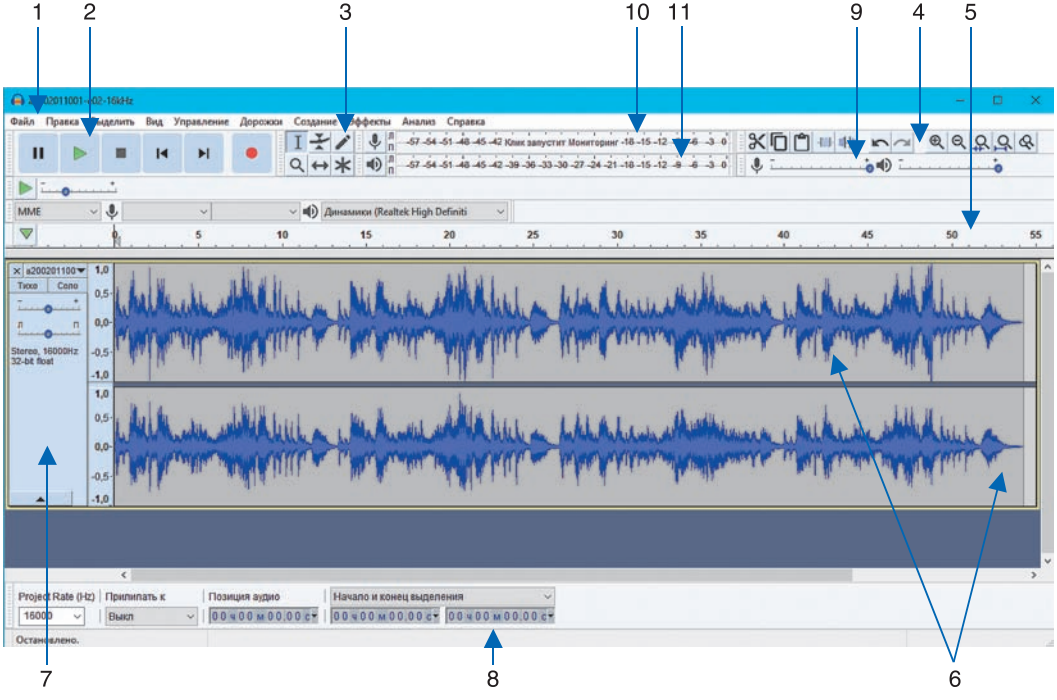
1. Вставьте номера страниц внизу страницы, выравнивание — от центра.
2. Установите для документа следующие поля: левое — 25 мм, правое — 10 мм, верхнее и нижнее — 15 мм.
3. Вставьте титульный лист, созданный в задании 2. Замените название.
4. Добавьте в верхний колонтитул свою фамилию. Установите параметр **Особый колонтитул для первой страницы.**

5*. Проиллюстрируйте документ.

ПРИЛОЖЕНИЯ

Приложение 1

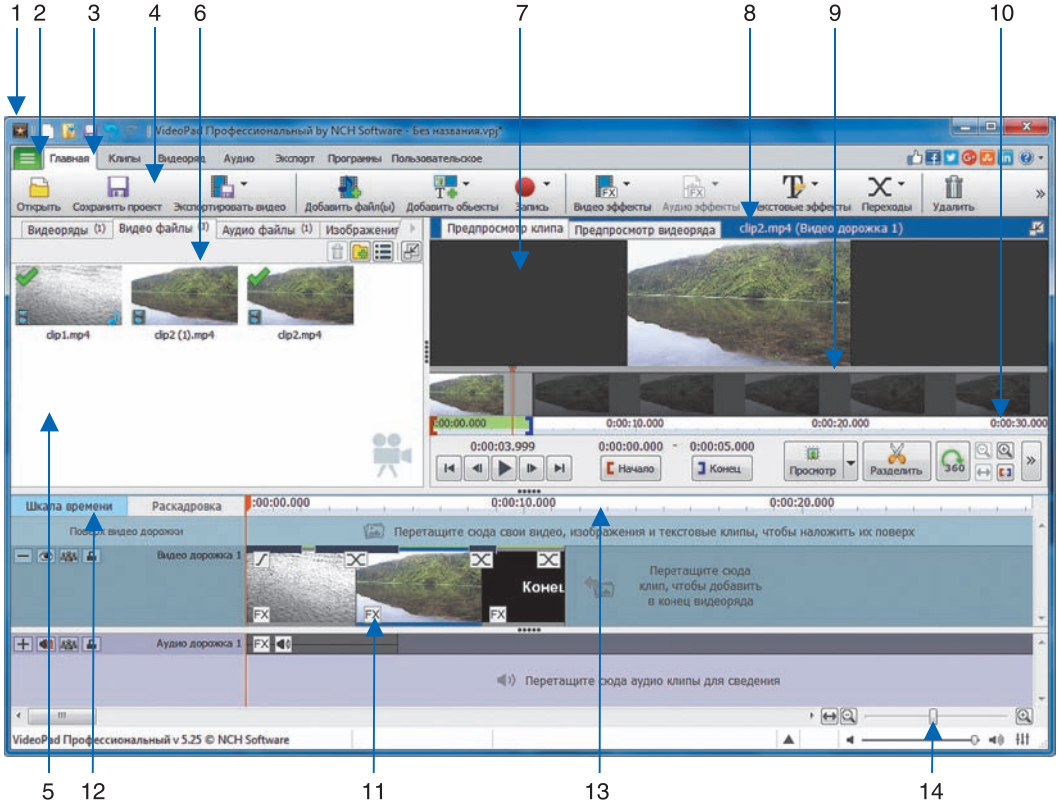
Элементы интерфейса аудиоредактора Audacity



1. Главное меню.
2. Панель воспроизведения и записи.
3. Панель инструментов.
4. Панель редактирования.
5. Шкала времени.
6. Дорожки.
7. Панель управления дорожками.
8. Панель выделения фрагментов.
9. Микшер.
10. Панель мониторинга уровня записи.
11. Панель мониторинга уровня воспроизведения.

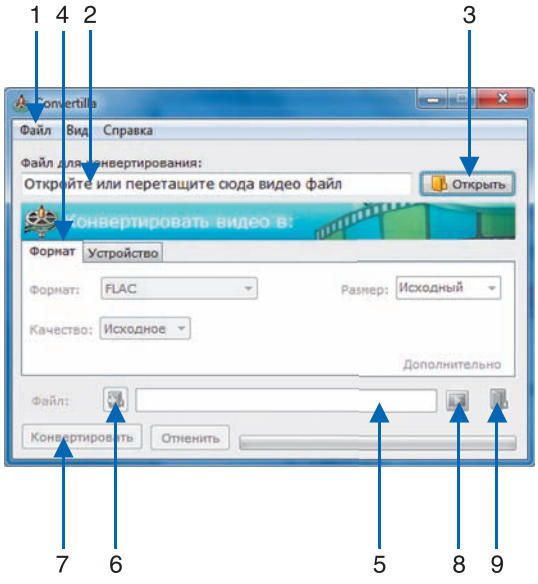
Панели можно перемещать.

Элементы интерфейса видеоредактора VideoPad



1. Панель быстрого доступа.
2. Кнопка вызова дополнительного меню.
3. Строка с названиями вкладок.
4. Панель инструментов выбранной вкладки.
5. Окно разделов с файлами.
6. Вкладки разделов с файлами.
7. Окно предпросмотра.
8. Заголовок окна предпросмотра с двумя вкладками.
9. Полоса эскизов.
10. Шкала времени в окне предпросмотра.
11. Окно видеоряда.
12. Кнопки выбора режима окна видеоряда.
13. Шкала времени в окне видеоряда.
14. Инструмент масштабирования изображения в окне видеоряда.

Элементы интерфейса видеоконвертера Convertilla



1. Меню.
2. Поле для отображения полного имени исходного файла.
3. Кнопка вызова окна **Выбор файла видео**.
4. Вкладка **Формат** для ввода параметров выходного файла.
5. Поле для отображения полного имени выходного файла.
6. Кнопка вызова окна **Открыть** для ввода имени выходного файла и выбора папки.
7. Кнопка запуска конвертации.
8. Кнопка запуска проигрывания выходного файла.
9. Кнопка открытия папки с выходным файлом.

Возможности вкладки Формат

Если загружен аудиофайл:

- на вкладке можно выбрать обозначение аудиоформата и значение параметра качества выходного файла;
- ссылка **Дополнительно** не работает.

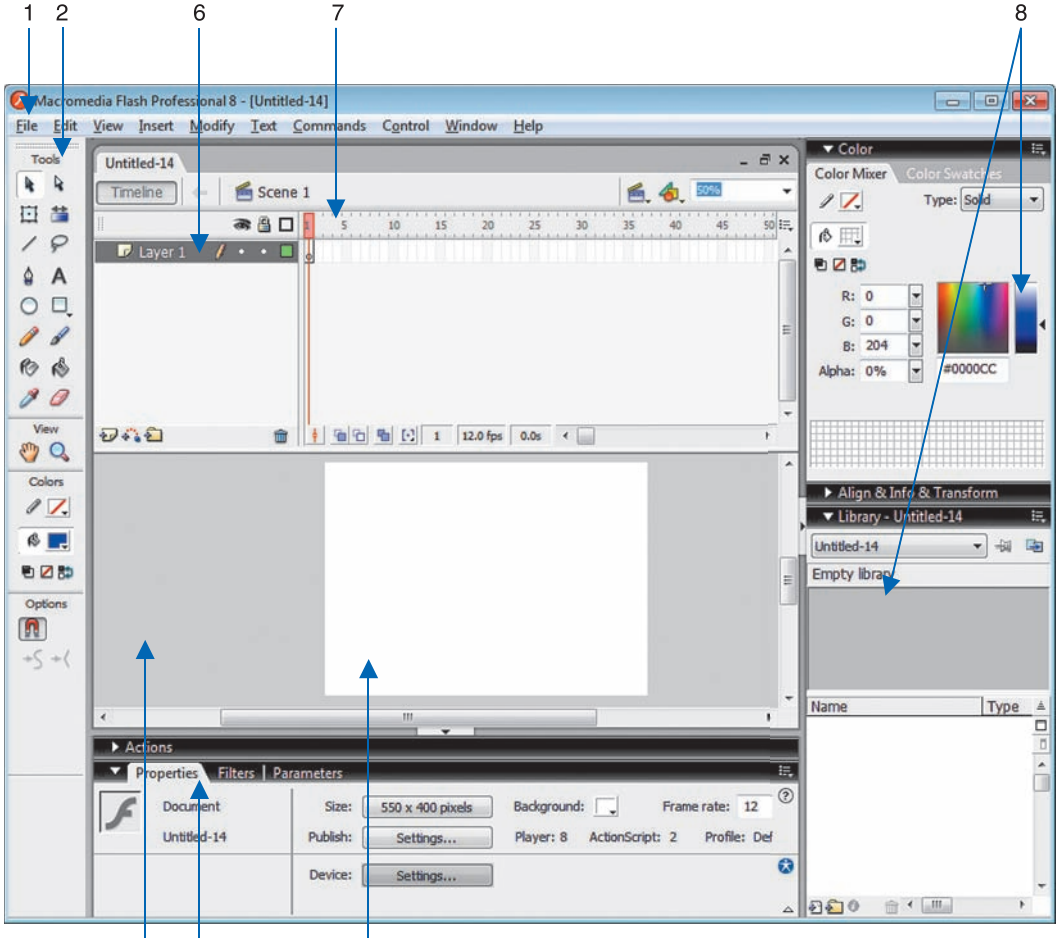
Если загружен видеофайл:

- на вкладке можно выбрать обозначение видеоформата, значение параметра качества и разрешение (в поле **Размер**);
- ссылка **Дополнительно** на вкладке открывает меню с параметрами **Без звука** и **Сохранить соотношение сторон**;
- выбор аудиокодека для видеофайла не предусмотрен.

Приложение 2

Элементы интерфейса редактора Flash

Окно редактора



1. Строка меню.
2. Панель инструментов.
3. Рабочая область.
4. Монтажный стол.
5. Панель свойств.
6. Список слоев.
7. Шкала времени.
8. Дополнительные панели.

Стартовая страница

По умолчанию после загрузки редактора Flash открывается стартовая страница, которая предоставляет доступ к трем разделам.

Чтобы открыть уже существующий файл, в разделе **Open a Recent Item** Стартовой страницы необходимо выбрать имя файла или папки, в которой он находится. Для создания нового фильма нужно выбрать **Flash Document** в разделе **Create New**. В третьем разделе находится список шаблонов.

Развернуть и свернуть дополнительные панели можно с помощью меню **Окно (Window)** или нажав на значок раскрывающегося списка (или) рядом с названием панели.

Панель свойств (Properties) отображает и позволяет редактировать свойства либо выделенных объектов, либо рабочей области, либо активного инструмента. Разворачивается и сворачивается при нажатии на значок раскрывающегося списка рядом со словом Properties.

Приложение 3

Математические функции языка Pascal

Запись на Pascal	Описание
`abs(x)`	Находит модуль числа x.
`ceil(x)`	Находит наименьшее целое $\geqslant x$ (`real`). Результат — число типа `integer`.
`cos(x)`	Вычисляет косинус числа x. Число x задается в радианах.
`DegToRad(x)`	Переводит градусы в радианы.
`floor(x)`	Находит наибольшее целое $\leqslant x$ (`real`). Результат — число типа `integer`.
`frac(x)`	Находит дробную часть действительного числа x (`real`). Результат — число типа `real`.
`int(x)`	Находит целую часть действительного числа x (`real`). Результат — число типа `real`.
`Max(a,b)`	Находит максимальное из чисел a и b.
`Min(a,b)`	Находит минимальное из чисел a и b.
`Odd(i)`	Логическая функция, которая возвращает значение True, если i нечетно, и False в противном случае.
`Power(x, y)`	Находит значение x^y (x, y — `real`). Результат — число типа `real`.
`RadToDeg(x)`	Переводит радианы в градусы.
`round(x)`	Округляет число x до ближайшего целого. Если число находится посередине между двумя целыми, то округляется к ближайшему четному (банковское округление): `round(2.5) = 2`, `round(3.5)= 4`.
`sin(x)`	Вычисляет синус числа x. Число x задается в радианах.
`sqr(x)`	Возводит число x в квадрат.
`sqrt(x)`	Находит корень квадратный из числа x. Результат — всегда число типа `real`.
`tan(x)`	Вычисляет тангенс числа x. Число x задается в радианах.
`trunc(x)`	Находит целую часть действительного числа x (`real`). Результат — число типа `integer`.

Некоторые графические примитивы

Запись на Pascal	Описание
`SetPixel(x,y,c)`	Закрашивает пиксел с координатами (x, y) цветом c.
`MoveTo(x,y)`	Устанавливает текущую позицию рисования в точку (x, y).
`LineTo(x,y)`	Рисует отрезок от текущей позиции до точки (x, y). Текущая позиция переносится в точку (x, y).
`Line(x1,y1,x2,y2)`	Рисует отрезок от точки $(x1, y1)$ до точки $(x2, y2)$.
`Line(x1,y1,x2,y2,c)`	Рисует отрезок от точки $(x1, y1)$ до точки $(x2, y2)$ цветом c.
`Circle(x,y,r)`	Рисует заполненный круг с центром (x, y) и радиусом r.
`Ellipse(x1,y1,x2,y2)`	Рисует заполненный эллипс, ограниченный прямоугольником, заданным координатами противоположных вершин $(x1, y1)$ и $(x2, y2)$.
`Rectangle(x1,y1,x2,y2)`	Рисует заполненный прямоугольник, заданный координатами противоположных вершин $(x1, y1)$ и $(x2, y2)$.
`RoundRect(x1,y1,x2,y2,w,h)`	Рисует заполненный прямоугольник со скругленными краями; $(x1, y1)$ и $(x2, y2)$ задают пару противоположных вершин, а w и h — ширину и высоту эллипса, используемого для скругления краев.
`Arc(x,y,r,a1,a2)`	Рисует дугу окружности с центром в точке (x, y) и радиусом r, заключенную между двумя лучами, образующими углы $a1$ и $a2$ с осью OX ($a1$ и $a2$ — вещественные, задаются в градусах и отсчитываются против часовой стрелки).
`Pie(x,y,r,a1,a2)`	Рисует заполненный сектор круга с центром в точке (x, y) и радиусом r, заключенный между двумя лучами, образующими углы $a1$ и $a2$ с осью OX ($a1$ и $a2$ — вещественные, задаются в градусах и отсчитываются против часовой стрелки).
`TextOut(x,y,z)`	Выводит строку или число z в прямоугольник с координатами левого верхнего угла (x, y).
`DrawTextCentered(x,y,x1,y1,z)`	Выводит строку или число z, отцентрированную в прямоугольнике с координатами $(x, y, x1, y1)$.
`FloodFill(x,y,c)`	Заливает область одного цвета цветом c, начиная с точки (x, y).

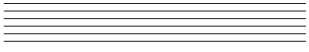
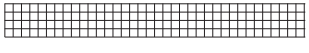
Стили пера (Pen)

Запись на Pascal	Описание
psSolid	Сплошное перо (по умолчанию)
psClear	Прозрачное перо
psDash	Штриховое перо
psDot	Пунктирное перо
psDashDot	Штрихпунктирное перо
psDashDotDot	Альтернативное штрихпунктирное перо (штрих два пунктира)

Стили кисти (Brush)

Запись на Pascal	Описание
bsSolid	Сплошная кисть (по умолчанию)
bsClear	Прозрачная кисть
bsHatch	Штриховая кисть
bsGradient	Градиентная кисть
Для всех кистей используется свойство Color. Для штриховой кисти дополнительно можно устанавливать свойства Hatch и HatchBackgroundColor, для градиентной — свойство GradientSecondColor.	

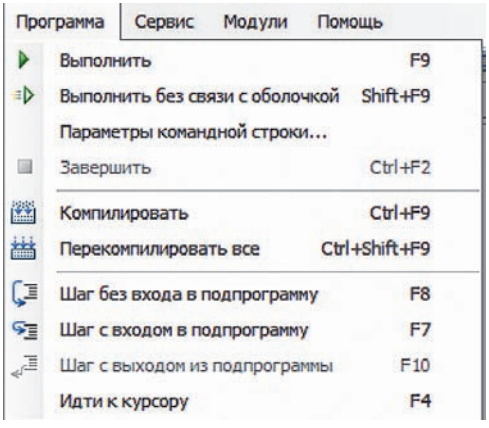
Некоторые стили штриховки кисти (HatchStyle)

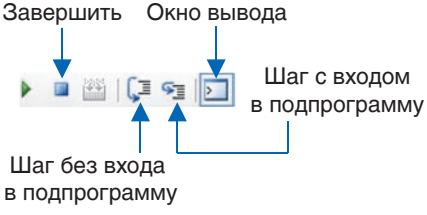
Запись на Pascal	Описание
bhHorizontal	
bhVertical	
bhForwardDiagonal	
bhCross	
bhZigZag	

Отладка программ в среде PascalABC

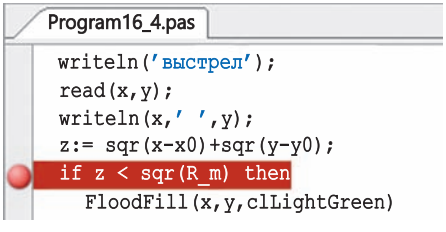
Программа может содержать логические ошибки, которые позволяют выполнить программу, однако результат выполнения будет отличаться от ожидаемого. Наличие таких ошибок определяется с помощью тестирования, а исправлять их позволяет отладчик, входящий в состав среды программирования PascalABC. Он позволяет выполнять программу, наблюдая за изменением значений переменных.

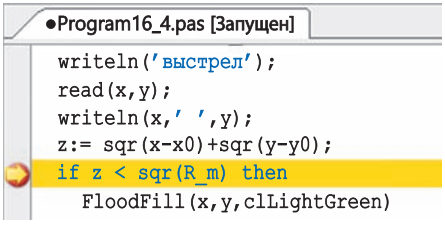
Команды отладчика собраны в меню **Программа** и вынесены на **Панель инструментов**.





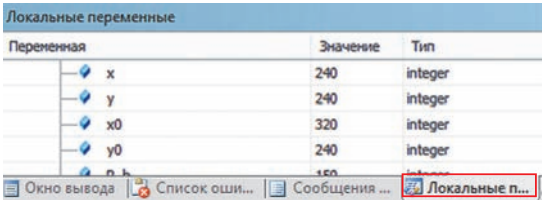
Некоторые строки программы могут быть помечены как **точки прерывания**. Для этого достаточно кликнуть мышью слева от строки программы. Когда в процессе выполнения программы достигается точка прерывания, выполнение программы приостанавливается. Строка, на которой остановилось выполнение программы, подсвечивается желтым.

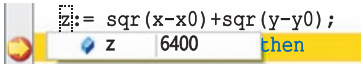




После этого можно просмотреть значения переменных в окне вывода на вкладке **Локальные переменные**, начать пошаговое выполнение программы или выполнить программу до следующей точки прерывания. В режиме

отладки, наведя указатель мыши на имя переменной, в окне редактора кода можно сразу увидеть ее значение.





Команды отладки		
Команда	Функция	Клавиша
Выполнить	Выполняет программу	F9
Завершить	Завершает сеанс отладки или выполнение программы	Ctrl+F2
Идти к курсору	Выполняет программу до строки, на которой установлен курсор	F4
Шаг без входа в подпрограмму	Выполняет строку программы. При наличии вызова подпрограммы в этой строке выполняет подпрограмму полностью	F8
Шаг с входом в подпрограмму	Выполняет строку программы. При наличии вызова подпрограммы в этой строке переходит к выполнению команд подпрограммы	F8
Шаг с выходом из подпрограммы	Выполняет подпрограмму до конца и передает управление на команду, следующую за вызовом данной подпрограммы в основной программе	F10

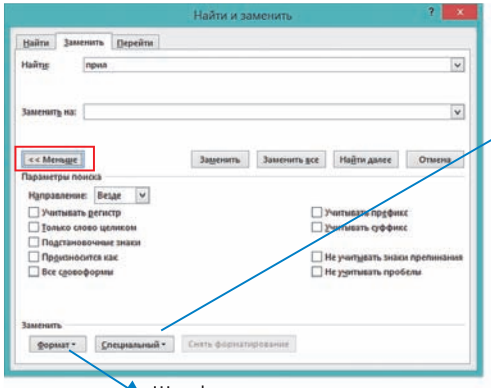
Приложение 4

Работа с текстовым документом

Поиск и замена

В окне **Заменить** есть кнопка **Больше**, нажав на которую получим дополнительные возможности замены в тексте. После нажатия надпись на кнопке меняется на **Меньше**.

Кнопка **Формат** позволяет указать параметры форматирования текста, которые нужно учитывать при поиске и замене.



Шрифт
Абзац
Табуляция...
Язык
Рамка...
Стиль...
Выделение цветом

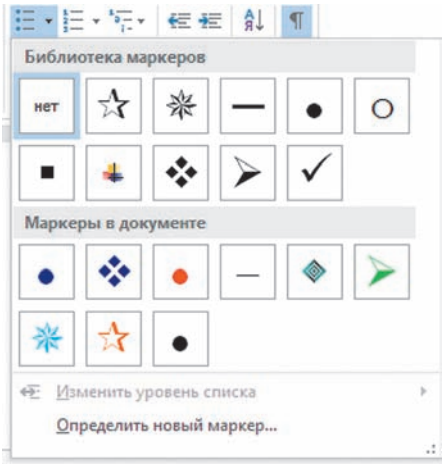
Знак абзаца
Знак табуляции
Любой знак
Любая цифра
Любая буква
Знак крышки
§ Конец раздела
¶Конец абзаца
Разрыв столбца
Длинное тире
Короткое тире
Знак концевой сноски
Поле
Знак сноски
Графический объект
Разрыв строки
Разрыв страницы
Неразрывный дефис
Неразрывный пробел
Мягкий перенос
Разрыв раздела
Пустое пространство

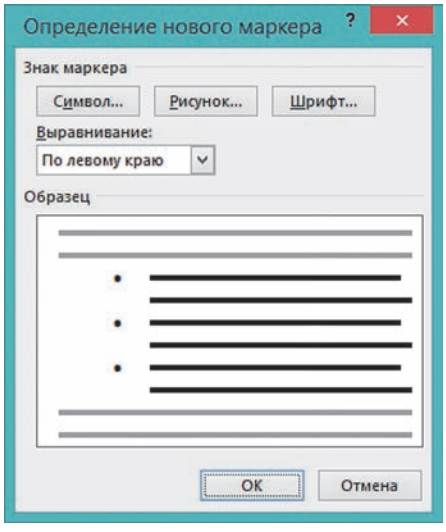
Кнопка **Специальный** позволяет искать и заменять непечатные символы: символы абзаца, разрыва строки, переноса и др.

Создание списков

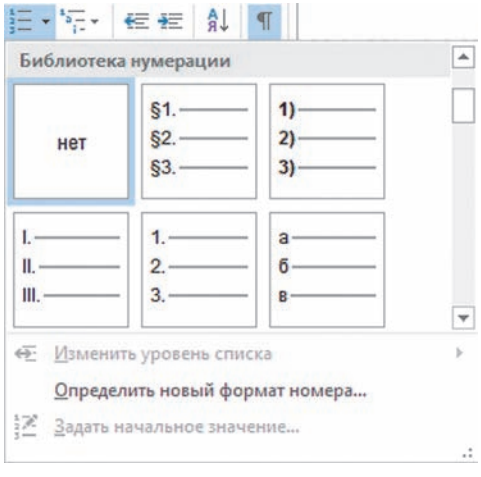
При создании списков можно не только использовать для них готовые стили, но и создавать свои.

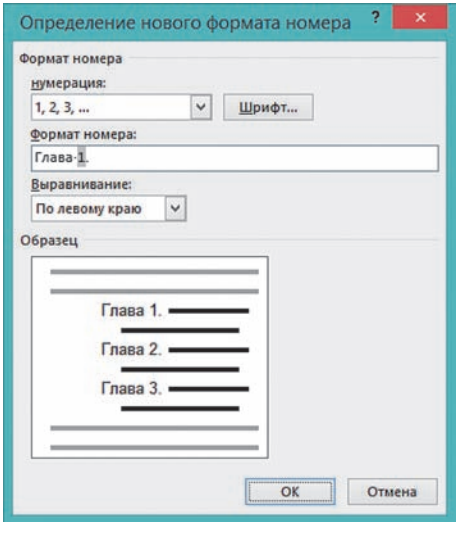
Команда **Определить новый маркер** открывает соответствующее окно, в котором в качестве маркера можно определить любой символ из таблицы символов или произвольный рисунок. С помощью кнопки **Шрифт** можно задать параметры символа: цвет, размер, начертание.





Для нумерованных списков можно определять свой формат номера с помощью команды **Определить новый формат номера**. В строке **Формат номера** перед номером можно вписать любое слово (глава, пример и т. п.), которое будет приписываться перед номером в выбранном списке.





При создании списков придерживаются следующих правил:

1. Предложение перед списком может оканчиваться двоеточием или точкой. Двоеточие ставится, если в этом предложении содержится слово или словосочетание, указывающее на то, что далее последует список или спи-

сок разъясняет то, о чем говорится в предшествующем ему предложении. В противном случае перед списком ставится точка.

2. В списках, в которых для нумерации используют числа (арабские или римские), текст начинается с прописной буквы, если после номера стоит точка, и со строчной буквы, если после номера стоит скобка.

3. В списках, в которых для нумерации используют буквы (русские или латинские), текст начинается с прописной буквы, если используется прописная буква с точкой, и со строчной буквы, если для нумерации используются строчные буквы со скобкой.

4. В маркированных списках текст начинается со строчной буквы.

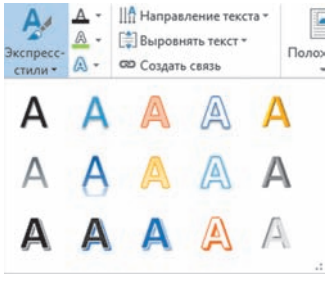
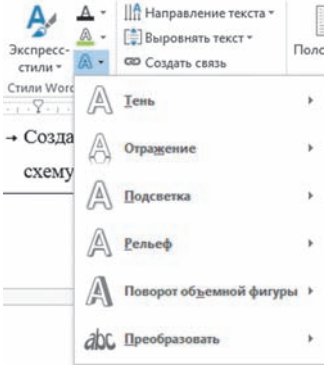
5. После элемента списка может стоять:

- запятая, если элементом списка является одно слово;
- точка с запятой, если элемент списка начинался со строчной буквы;
- точка, если элемент списка начинался с прописной буквы.

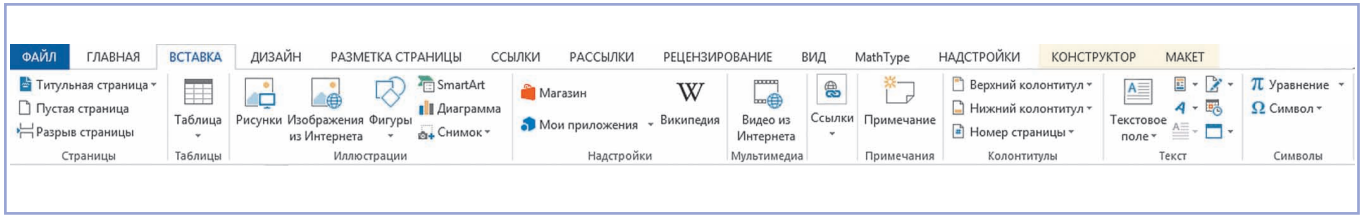
6. В конце списка ставится точка.

7. При использовании списков следует обязательно обращать внимание на то, чтобы начальные слова каждого элемента списка были согласованы между собой в роде, числе и падеже.

Параметры WordArt

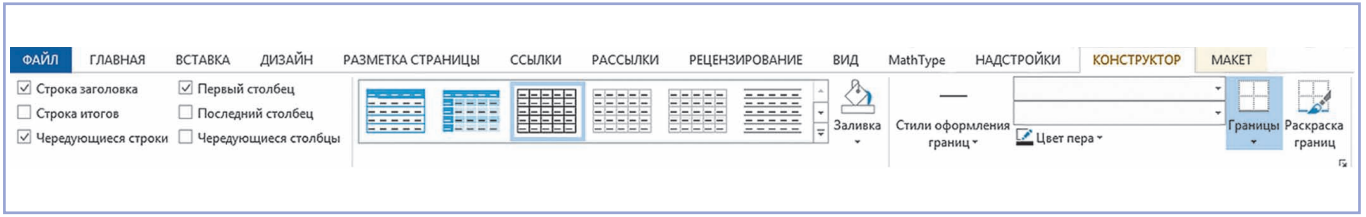
Стили WordArt	Эффекты WordArt	Команда **Преобразовать**
		

Вкладка Вставка

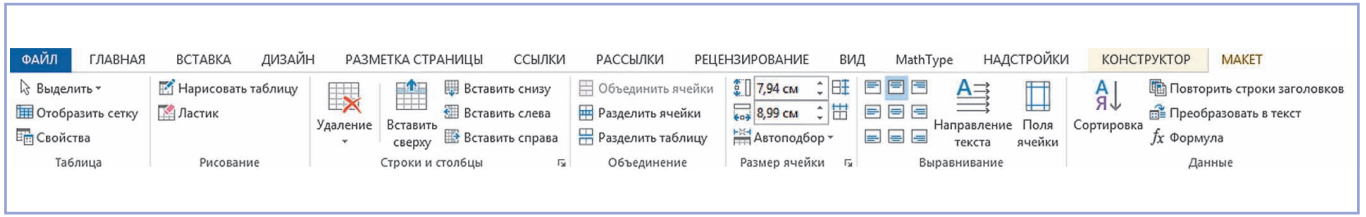


Вкладка Работа с таблицами

Конструктор



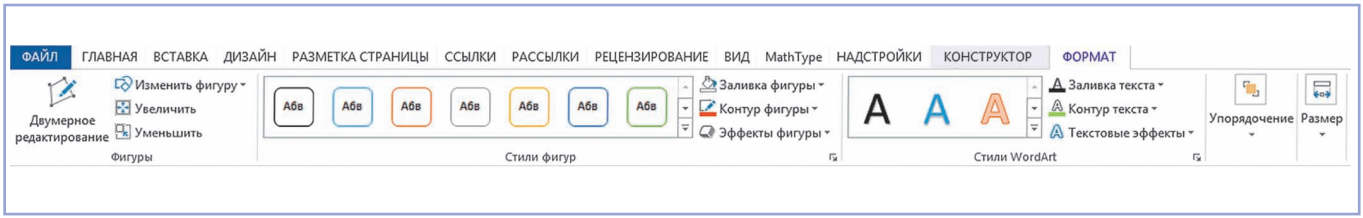
Макет



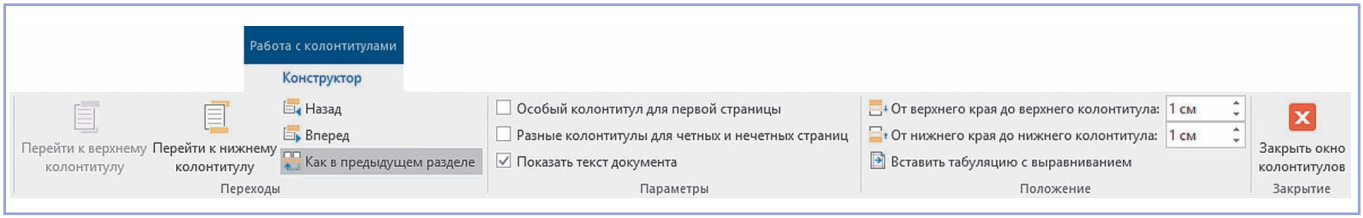
Параметры SmartArt

Конструктор

Формат



Вкладка Работа с колонтитулами



(Название и номер учреждения образования)

Учебный год	Имя и фамилия учащегося	Состояние учебного пособия при получении	Оценка учащемуся за пользование учебным пособием
20 /			
20 /			
20 /			
20 /			
20 /			
20 /			

Учебное издание

Котов Владимир Михайлович
Лапо Анжелика Ивановна
Быкадоров Юрий Александрович
Войтехович Елена Николаевна

ИНФОРМАТИКА

Учебное пособие для 8 класса
учреждений общего среднего образования с русским языком обучения

Зав. редакцией *Г. А. Бабаева*. Редактор *Е. И. Даниленко*. Художественные редакторы *А. Н. Богушевич, О. Н. Карпович*. Обложка *А. Н. Богушевича*. Техническое редактирование и компьютерная верстка *И. И. Дубровской*. Корректоры *В. С. Бабеня, Е. П. Тхир, А. В. Алешко*.

Подписано в печать 13.08.2018. Формат $70\times90\,{}^{1}/_{16}$. Бумага офсетная. Гарнитура школьная. Печать офсетная. Усл. печ. л. 12,29. Уч.-изд. л. 10,0. Тираж 116 000 экз. Заказ 719.

Издательское республиканское унитарное предприятие «Народная асвета» Министерства информации Республики Беларусь. Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/2 от 08.07.2013. Пр. Победителей, 11, 220004, Минск, Республика Беларусь.

ОАО «Полиграфкомбинат им. Я. Коласа». Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 2/3 от 04.10.2013. Ул. Корженевского, 20, 220024, Минск, Республика Беларусь.