Н. В. Морзе, О. В. Барна, В. П. Вембер

ИНФОРМАТИКА

Учебник для 8 класса общеобразовательных учебных заведений с обучением на русском языке

Рекомендовано Министерством образования и науки Украины

Киев
Оріон
2016

УДК 004(075.3)
ББК 32.973я721
М79

Рекомендовано Міністерством освіти і науки України
(*наказ МОН України від 10.05. 2016 р. № 491*)

ВИДАНО ЗА РАХУНОК ДЕРЖАВНИХ КОШТІВ. ПРОДАЖ ЗАБОРОНЕНО

Експерти, які здійснювали експертизу даного підручника під час проведення конкурсного відбору проектів підручників для учнів 8 класу загальноосвітніх навчальних закладів і зробили висновок про доцільність надання підручнику грифа «Рекомендовано Міністерством освіти і науки України»:

Франчук П. А., учитель Ямпільської загальноосвітньої школи I — III ступенів, старший учитель;

Довганюк А. Ф., методист із навчальних дисциплін Новоселицького районного методичного комітету Чернівецької області, учитель-методист;

Погромська Г. С., доцент кафедри прикладних математики, механіки та інформатики Миколаївського національного університету ім. В. О. Сухомлинського, кандидат педагогічних наук.

Перекладено за виданням:

Інформатика : підруч. для 8 кл. загальноосвіт. навч. закладів / Н. В. Морзе, О. В. Барна, В. П. Вембер. — К. : УОВЦ «Оріон», 2016. — 240 с.

Навчальне видання

МОРЗЕ Наталія Вікторівна
БАРНА Ольга Василівна
ВЕМБЕР Вікторія Павлівна

ІНФОРМАТИКА
Підручник для 8 класу загальноосвітніх навчальних закладів з навчанням російською мовою

(Російською мовою)

Редактор *В. М. Ліченко*
Коректор *С. В. Войтенко*
Головний художник *І. П. Медведовська*
Технічний редактор *Е. А. Авраменко*

У підручнику використано світлини та ілюстрації Federico Caputo, Halina Przeszzlo, Grzegorz Kula, Jiri Hera, Alexander Podshivalov, Roman Kunitski, Maksym Zaletskyy, Eric Isselée.

Формат 84x108 $^{1}/_{16}$.
Ум. друк. арк. 25,2 + 0,42 форзац.
Обл.-вид. арк. 23,50 + 0,40 форзац.
Наклад 2748 пр.
Зам. №

ТОВ «Український освітянський видавничий центр “Оріон”»

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції
Серія ДК № 4918 від 17.06.2015 р.
Адреса видавництва: 03061, м. Київ, вул. Миколи Шепелєва, 2
www.orioncentr.com.ua

Віддруковано
ТОВ «НВП Поліграфсервіс»,
вул. Юрія Коцюбинського, 4, к. 25,
м. Київ, 04053
Свідоцтво суб'єкта видавничої справи
серія ДК № 3751 від 01.04.2010.

Морзе Н. В.
М79 Информатика : учебник для 8 кл. общеобразоват. учебн. заведений с обучением на русском языке / Н. В. Морзе, О. В. Барна, В. П. Вембер. — К. : УОИЦ «Оріон», 2016. — 240 с. : илл.

ISBN 978-617-7355-65-5.

УДК 004(075.3)
ББК 32.973я721

ISBN 978-617-7355-45-7 (укр.)
ISBN 978-617-7355-65-5 (рус.)

ДОРОГИЕ ВОСЬМИКЛАССНИКИ!

Этот учебник поможет вам в изучении информатики. Он состоит из семи разделов, содержащих темы, с которыми вы будете знакомиться на одном, двух или более уроках. Вы узнаете, как кодируются данные с помощью компьютера, научитесь различать составляющие компьютера, устройства и программы. Продолжите совершенствовать свои знания и умения по работе с текстовыми данными в среде текстового процессора и числовыми данными в среде табличного процессора. Научитесь использовать новые программы для работы с объектами мультимедиа. На страницах учебника вы найдёте описание языков программирования *Python* и *Free Pascal*, примеры программ, а с помощью сред программирования, поддерживающих эти языки, научитесь разрабатывать различные программные проекты для обучения и развлечений. В учебнике много заданий и упражнений, в том числе — компетентностных задач и проектов.

Готовясь к уроку, обратите внимание на перечень вопросов, которые вы изучали на уроках информатики ранее. Эти вопросы предложены в рубрике

ВСПОМНИТЕ:

Вспомните изученное, просмотрите записи в рабочих тетрадях или материалы в Интернете и будьте готовы использовать свои знания при изучении новых тем, приобретая новые умения и компетентности.

Выстраивайте собственную стратегию обучения, выполняйте рефлексию и оценивание, используя рубрику

ВЫ УЗНАЕТЕ:

В этой рубрике указаны важнейшие вопросы, которые рассматриваются в теме.

В учебнике вы также будете использовать следующие рубрики.

ИЗУЧАЕМ

Материалы рубрики **Изучаем** целесообразно читать накануне урока, чтобы вместе с одноклассниками и учителем обсудить непонятное и поделиться выученным.

Важнейшие определения по этой теме выделены и обозначены так.

ДЕЙСТВУЕМ

Выполняя упражнения из рубрики **Действуем**, вы научитесь кодировать данные, создавать мультимедийные проекты, работать с текстовыми и числовыми данными, разрабатывать в средах программирования различные проекты.

Рубрика **Исследуем** включает задания на проведение экспериментов, самостоятельный поиск ответов, новых возможностей.

«Фаворитные» задания, отмеченные значками   , предназначены для тех, кто в будущем хочет быть успешным.

Рубрика **Обсуждаем** предлагает вам список вопросов, которые можно обсудить коллективно или в малых группах. Их можно найти в папке *Обсуждаем* или по ссылке, представленной соответствующим QR-кодом.

Уверены, вы научитесь планировать свою учебную деятельность, мыслить логически, искать нестандартные решения задач, делать выводы и самостоятельно принимать решения. Этому помогут упражнения из рубрики **Работаем самостоятельно**.

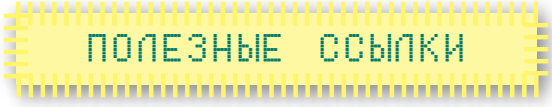
Также важно научиться аргументировать свои мысли при обсуждении вопросов в парах и группах или признавать ошибочность своих рассуждений в пользу более логичных и доказательных. Для этого предназначены задания рубрики **Работаем в парах.**

Сотрудничество, умение учиться друг у друга, поддержка в практической деятельности — залог успеха в будущем.

Не оставляйте без внимания закладки 

, ведь изучение информатики способствует всестороннему развитию, формированию умения учиться самостоятельно.

Обязательно познакомьтесь со ссылками, записанными в закладке



Быстро ввести нужный адрес можно с помощью QR-кода. Здесь вы получите много полезных сведений и инструментов для обучения, доступных в сети Интернет.

Желаем вам успеха!

РАЗДЕЛ 1

КОДИРОВАНИЕ ДАННЫХ

1. КОДИРОВАНИЕ ДАННЫХ

ВСПОМНИТЕ:

- как связаны между собой понятия *информация*, *сообщение*, *данные*;
- способы представления сообщений;
- особенности основных информационных процессов: поиска, передачи, обработки, хранения сообщений;
- особенности обработки сообщений.

ВЫ УЗНАЕТЕ:

- что такое кодирование сообщений;
- каковы особенности двоичного кодирования;
- как кодируются в компьютере текстовые сообщения;
- для чего используются таблицы кодировки символов;
- как вычислить длину двоичного кода сообщения.

ИЗУЧАЕМ

1. Что такое кодирование сообщений?

Вы знаете, что информация передаётся с помощью сообщений, которые можно представить текстовым, графическим, звуковым способами, а также видео, жестами и несколькими способами одновременно — комбинированным сообщением.

Для передачи устных сообщений мы используем язык. А любое письменное сообщение состоит из набора разных знаков (символов) определённого алфавита. Например, для передачи числовых данных используют алфавит, состоящий из 11 знаков: цифр 0–9 и десятичной запятой для отделения целой и дробной части в десятичной дроби.

Люди постоянно получают, ищут, хранят, обрабатывают и передают сообщения. При этом они часто применяют различные устройства. Использование устройств определяет необходимость передачи сообщений с помощью специальных знаков.

Код — это система правил для преобразования сообщений, содержащих текст, звук, изображение, жесты и т. п.

Код Морзе	Штрихкод	QR-код
...———...	4 820123 456784	

Кодирование сообщений — это процесс преобразования по определённым правилам одного вида набора данных в другой. При кодировании сообщения происходит изменение его вида без изменения содержания.

Рис. 1.1

В процессе обмена сообщениями мы осуществляем две операции: кодирование и декодирование. Первая связана с переходом от исходной формы вида сообщения к форме, удобной для хранения, передачи или обработки. А вторая — с обратным переходом к исходному виду сообщения.

Процесс получения и передачи сообщений можно изобразить в виде схемы (рис. 1.1).

Кодирование и декодирование относятся к процессу обработки сообщений, осуществляемых с помощью компьютера.

В последнее время всё чаще для кодирования данных стали использовать QR-коды.

QR-код — квадратная картинка, в которой закодированы некоторые сведения:

- обычный текст;
- адрес веб-страницы в Интернете;
- номер телефона;
- координаты места на карте;
- персональная визитная карточка и т. п.

Выглядит QR-код как квадратное изображение, в углах которого обычно расположены три больших квадрата. Они служат своеобразными ориентирами для программ-считывателей. С их помощью определяется уровень наклона и деформация пропорций. Остальные фигуры — это, собственно, и есть зашифрованные сведения. В отличие от классического штрихкода, QR-код может включать в себя достаточно большие объёмы данных.

Специальный вид QR-кодов обеспечивает считывание и декодирование данных с помощью современных устройств, оборудованных камерами, например мобильных телефонов. Достаточно направить камеру телефона на код, и сразу можно получить доступ к его содержимому.

С появлением мощных мобильных телефонов, оборудованных встроенными камерами, QR-коды стали широко использовать в мире.

Для считывания QR-кодов с помощью мобильного телефона на нём нужно установить специальную программу (рис. 1.2).

QR-коды создают с помощью специальных программ, которые можно загрузить на компьютер, или используют специальные сайты в Интернете. Например, с помощью онлайн-сервиса для генерирования QR-кода создан QR-код его адреса *http://ru.qr-code-generator.com/*

Интересно

QR-коды (от англ. *Quick Response* — быстрая реакция, быстрый ответ) были разработаны в 1994 г. японской компанией DensoWave. В Японии QR-коды стали активно распространяться ещё в начале 2000-х годов: их размещают на упаковке товаров, печатают в буклетах, используют в рекламе, справочниках, играх и т. п.

Рис. 1.2

2. Каковы особенности двоичного кодирования?

Сообщения могут кодироваться разными способами. Выбор способа кодирования зависит от вида сообщения, которое нужно закодировать: текст, число, графическое изображение, звук или видео. Для обработки с помощью компьютера сообщения представляются (кодируются) в виде последовательностей электрических или магнитных сигналов двух видов. Каждый сигнал одного вида условно обозначают цифрой 0, а другого вида — 1.

Кодирование сообщений с использованием двух сигналов называется **двоичным**. Набор данных, полученный в результате двоичного кодирования, называется **двоичным кодом.**

Цифра 0 или 1 в двоичном коде сообщения называется **битом** (англ. *binary digit* — двоичная цифра).

Одной из двух цифр 0 или 1 можно закодировать, например:

- правильность утверждения: неправильно (0) или правильно (1);
- состояние выключателя: выключен (0) или включён (1) и т. п.

Из двух битов можно составить 4 ($4 = 2^2$) кода (00, 01, 10 и 11). Ими можно закодировать, например, четыре четверти координатной плоскости: 00 — левая верхняя; 01 — правая верхняя; 10 — левая нижняя; 11 — правая нижняя.

Из трёх битов можно составить уже 8 ($8 = 2^3$) кодов (000, 001, 010, 011, 100, 101, 110, 111). Ими можно закодировать, например, стороны горизонта (рис. 1.3). Из четырёх битов можно составить $2^4 = 16$ кодов, из пяти — $2^5 = 32$ кода и т. п. Из восьми битов можно составить $2^8 = 256$ кодов, и этого количества кодов достаточно, чтобы закодировать все буквы английского и русского (или любого другого) алфавитов, арабские цифры, знаки препинания, знаки арифметических действий, а также некоторые другие символы.

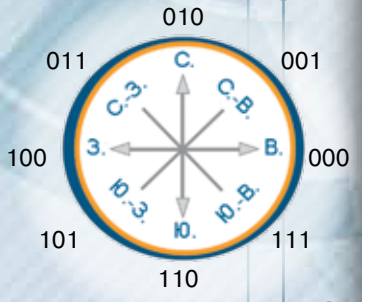


Рис. 1.3

Последовательность из восьми битов называется **байтом**.

1 байт = 8 бит

В таблице степеней числа 2 представлено количество разных сообщений, которые можно закодировать с помощью соответствующего количества битов:

i	0	1	2	3	4	5	6	7	8	9	10
$N = 2^i$	1	2	4	8	16	32	64	128	256	512	1024

3. Как кодируются в компьютере текстовые сообщения?

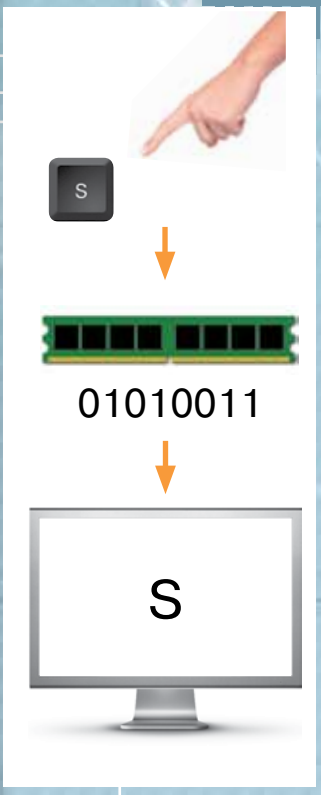
Традиционно для кодирования одного символа используется двоичный код, длина которого равна 1 байту, т. е. 1 символ = 1 байт = = 8 бит. Тогда можно рассчитать, какое количество разных символов

Интересно

С помощью формулы Хартли $N = 2^i$ можно определить возможное количество разных сообщений N, если i — количество битов в коде сообщения.

возможно закодировать: $N = 2^8 = 256$. Такое количество символов достаточно для представления текстовых сообщений, включая прописные и строчные буквы русского, украинского и латинского алфавитов, цифры, знаки, графические символы и т. п. Кодирование заключается в том, что каждому символу ставится в соответствие уникальный десятичный код от 0 до 255 или соответствующий ему двоичный код от 00000000 до 11111111.

Пользователь нажимает на клавиатуре клавишу с символом, и в компьютер поступает определённая последовательность из восьми электрических импульсов (двоичный код символа). Код символа хранится в оперативной памяти компьютера, где занимает один байт. В процессе вывода символа на экран компьютера происходит обратный процесс — декодирование, т. е. преобразование кода символа в его изображение.



Длина двоичного кода текстового сообщения — это количество битов или байтов в двоичном коде этого сообщения.

При таком кодировании одна строка текста этого учебника имеет среднюю длину двоичного кода приблизительно 50 байт, одна страница — приблизительно 2000 байт, а весь учебник (240 с.) — приблизительно 480 000 байт.

Для обозначения длины двоичного кода сообщений используют и бóльшие единицы измерения, названия которых, согласно Международной системе единиц (СИ), образуются с помощью префиксов *кило-*, *мега-*, *гига-*, *тера-* и т. д. Исторически сложилось так, что эти префиксы (*кило-*, *мега-*, *гига-*, *тера-*) в информатике трактовались иначе, не так, как, например, в математике, они основаны на степени числа 2, а именно:

1 Кб (килобайт) = 2^{10} байт = 1024 байт;

1 Мб (мегабайт) = 2^{10} Кб = 2^{20} байт = 1 048 576 байт;

1 Гб (гигабайт) = 2^{10} Мб = 2^{20} Кб = 2^{30} байт;

1 Тб (терабайт) = 2^{10} Гб = 2^{20} Мб = 2^{30} Кб = 2^{40} байт.

Чтобы перевести биты в байты, нужно число битов разделить на 8. Например: 32 бита — это 4 байта. Чтобы перевести байты в килобайты, нужно число байтов разделить на 1024. Например: в 2048 байтах будет 2 Кб. Аналогично можно перейти к другим единицам измерения.

Чтобы перевести байты в биты, нужно число байтов умножить на 8. Например: в 3 байтах будет 24 бита.

Чтобы перевести килобайты в байты, нужно число килобайтов умножить на 1024. Например: в 3 килобайтах будет 3072 байта и, соответственно, 24 576 бит.

ДЕЙСТВУЕМ

Упражнение 1. Длина двоичного кода текста.

Задание. В книге 150 страниц, на каждой странице — 40 строк, в каждой строке 60 символов (включая пробелы). Найдите длину двоичного кода текста книги, если для кодирования каждого символа использовано 8 бит.

Решение. Вычислим количество символов в книге:

$60 \cdot 40 \cdot 150 = 360\,000$ символов.

Поскольку длина двоичного кода 1 символа — 8 бит = 1 байт, длина двоичного кода книги равна 360 000 байт. Переведём байты в бóльшие единицы:

360 000 байт : 1024 = 351,56 Кб.

351,56 Кб : 1024 = 0,34 Мб.

Таблица 1.1

Код	Символ	Код	Символ	Код	Символ	Код	Символ
32	про-бел	56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	«	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60	<	84	T	108	i
37	%	61	=	85	U	109	m
38	&	62	>	86	V	110	n
39	‘	63	?	87	W	111	0
40	(	64	@	88	X	112	p
41	)	65	A	89	У	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	{
52	4	76	L	100	d	124	|
53	5	77	M	101	e	125	}
54	6	78	N	102	f	126	~
55	7	79	O	103	g	127	DEL

4. Для чего используются таблицы кодировки символов?

Нажатие клавиши на клавиатуре приводит к тому, что сигнал посылается в компьютер в виде двоичного числа, хранящегося в кодовой таблице. Кодовая таблица устанавливает соответствие между символами и их двоичными кодами для отображения текстовых данных в компьютере. Для того чтобы весь мир одинаково кодировал текстовые данные, нужны единые таблицы кодировки.

Во всём мире в качестве стандарта принята таблица ASCII (*American Standard Code for Information Interchange* — Американский стандартный код для обмена сообщениями). Созданная в 1963 г. система кодировки ASCII предусматривала кодирование 128 символов, коды которых состояли из 7 бит ($2^7 = 128$). Со временем кодировка расширилась до 256 символов ($2^8 = 256$), при этом коды первых 128 символов не изменились.

Таблица кодировки ASCII (табл. 1.1) состоит из базовой (значения кодов от 0 до 31) и расширенной (значения кодов от 32 до 255) таблиц. Коды с 33 по 127 являются интернациональными и соответствуют символам латинского алфавита, цифрам, знакам арифметических операций и знакам пунктуации. Коды с 128 по 255 являются национальными, т. е. в национальных кодировках одному и тому же коду соответствуют разные символы.

В мире существуют и другие системы кодировки. В различных таблицах кодировки одни и те же символы могут иметь разные коды. В последнее время среди таблиц кодировки, содержащих украинские и русские буквы, самыми распространёнными являются KOI8-U и Windows-1251. Длина кода каждого символа в них — 1 байт.

Широко распространён международный стандарт Unicode — *Unicode Consortium* (UTF 32, UTF 16 и UTF 8), согласно которому на каждый символ отводится не один байт, а два, т. е. 16 бит, поэтому с его помощью можно закодировать не 256, а $2^{16} = 65\,536$ разных символов.

Упражнение 2. Тексты в разных системах кодировки.
Задание. Рассчитайте длину двоичного кода в разных системах кодировки для текстов, представленных в ячейках электронной таблицы *Тексты*.

Решение.

1. В своей структуре папок создайте папку *Кодирование*.
2. В папке *Кодирование данных* откройте файл *Тексты*.
3. Введите в ячейку *B4* формулу вычисления количества символов в тексте: =LEN(A4) (=ДЛСТР(A4) для программы

	A	B	C	D	E
1			Код		
2	Пословицы	Количество символов	Windows-1251	ASCII	Unicode
3			8	8	16
4	Не говори — не могу, а говори — научусь!	40	320	320	640
5	Не святые горшки лепят.				
6	Мудрым никто не родился, а научился.				

рис. 1.4

с интерфейсом на русском языке). Убедитесь, что встроенная в табличный процессор функция правильно определяет количество символов.

4. Выполните вычисления в ячейках четвёртой строки так, чтобы получить результат, как на рисунке 1.4.
5. Выполните вычисления в ячейках пятой и шестой строк.
6. Сохраните файл с тем же именем в папке *Кодирование* своей структуры папок.

ПОЛЕЗНЫЕ ССЫЛКИ

Статья о системе кодировки в Википедии:

https://ru.wikipedia.org/wiki/ASCII

5. Как вычислить длину двоичного кода сообщения?

Чтобы вычислить длину двоичного кода сообщения, нужно количество символов в тексте умножить на количество битов, необходимых для кодирования одного символа. Например: двоичное число 01010111 занимает в памяти 8 бит. Если его записать в виде текста в кодировке ASCII, длина кода будет 8 байт, или 64 бита, поскольку каждый символ кодируется с помощью 8 бит. Длина двоичного кода этого же текста в кодировке Unicode будет равна 16 байт, или 128 бит.

Не следует забывать, что пробелы нужно считать символами, поскольку они также вводятся с клавиатуры, имеют код и хранятся в памяти.

ДЕЙСТВУЕМ

Упражнение 3. Перекодирование.

Задание. Автоматическое устройство перекодировало текстовое сообщение длиной 48 символов, изначально записанное в 7-битном коде ASCII, в 16-битный код Unicode. Определите, на сколько при этом увеличилась длина двоичного кода сообщения.

Решение. Изменение кодировки с 7 бит на 16 бит увеличивает длину кода каждого символа на 16 – 7 = 9 бит. Поскольку сообщение содержит 48 символов, то его объём увеличился на 48 · 9 = 432 бита. Переведём биты в байты: 432 : 8 = 54 байта.

Упражнение 4. Модем.

Задание. Определите, сколько времени модем, работающий со скоростью 1200 бит/с, будет передавать данные десяти страниц текста из 40 строк по 80 символов в строке.

Решение. Вычислим общее количество символов на странице. Это 40 · 80 = 3200 символов.

Так как в кодировке ASCII длина кода одного символа 1 байт, общая длина двоичного кода данных на странице — 3200 байт. Выразим 3200 байт в битах, поскольку скорость модема указана в битах за секунду. Получим 3200 байт = 25 600 бит.

Разделим 25 600 бит на 1200 бит/с, получим 21,33 с. Таким образом, 10 страниц текста будут переданы модемом за 213,3 с, или 3 мин 33,3 с.

ИССЛЕДУЕМ

Упражнение 4. Кодировка веб-страниц.

Задание. Проанализируйте, как изменятся текстовые данные на главной странице веб-портала *Библиотека украинской литературы* (*http://www.ukrlib.com.ua/*), если изменить таблицу кодировки, использованную при её создании.

1. Откройте окно браузера, например *GoogleChrome*. В поле адреса введите адрес веб-портала *Библиотека украинской литературы* (*http://www.ukrlib.com.ua/*). Посмотрите, правильно ли отображаются текстовые надписи на странице.
2. Нажмите кнопку настройки и управления браузером. Выберите в открывшемся меню команду *Дополнительные инструменты/Кодировка* (рис. 1.5).

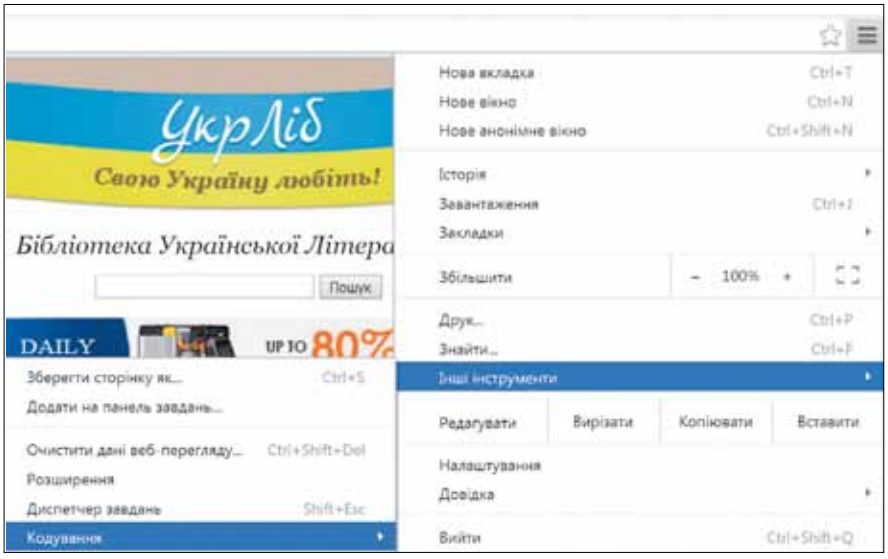


Рис. 1.5

3. Последовательно выбирайте таблицу кодировки из открывшегося меню. Сделайте вывод, какая из таблиц кодировки, кроме системы Windows-1251, правильно отображает содержимое веб-страницы.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 1* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Создайте в табличном процессоре таблицу перевода единиц длины двоичного кода сообщения в таком виде:

Количество символов двоичного кода, *бит*	байт	Кбайт	Мбайт	Гбайт	Тбайт

Предложите друг другу для использования и проверьте на практике другие таблицы перевода, например, килобайт в биты, байты, мегабайты, гигабайты, терабайты. Обсудите, как при этом будут изменяться формулы в ячейках электронной таблицы.

2. Подготовьте друг для друга текстовое сообщение, закодированное с помощью одной из таблиц кодировки. Один представляет придуманное сообщение, используя таблицу Unicode, другой — Windows-1251. Проверьте правильность кодирования, для этого обменяйтесь кодами и декодируйте сообщение. Определите длину двоичного кода каждого из сообщений. Сделайте вывод.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Определите длину двоичного кода слова из 24 символов в кодировке Unicode.
2. Длина двоичного кода текстового сообщения составляет 8192 бита. Выразите это значение в килобайтах.
3. Длина двоичного кода текстового сообщения составляет 2 097 152 байта. Выразите это значение в мегабайтах.
4. Длина двоичного кода текстового сообщения, подготовленного с помощью компьютера, составляет 3,5 Кб. Сколько символов содержит этот текст? Достаточно ли в задаче данных, чтобы можно было дать однозначный ответ?
5. Автоматическое устройство перекодировало на русский язык текстовое сообщение, изначально записанное в 16-битном коде Unicode, в 8-битный код KOI-8. При этом длина двоичного кода сообщения уменьшилась на 480 бит. Сколько символов содержит сообщение?
6. Известно, что на каждой странице документа 128 строк, по 48 символов каждая. Сколько страниц в документе, если длина его двоичного кода составляет 720 Кб, при условии, что каждый символ кодировался 2-байтовым кодом Unicode?

ПОЛЕЗНЫЕ ССЫЛКИ

Видеоурок «Кодирование текстовой информации»:
http://www.youtube.com/watch?v=3BDE0oxevUQ

2. ПРАКТИЧЕСКАЯ РАБОТА 1

РЕШЕНИЕ ЗАДАЧ НА ВЫЧИСЛЕНИЕ ДЛИНЫ ДВОИЧНОГО КОДА ТЕКСТОВЫХ ДАННЫХ

ВСПОМНИТЕ

- Как вычислить длину двоичного кода текстового сообщения.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 1.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

1. Длина двоичного кода текста, набранного на компьютере с использованием кодировки Unicode (каждый символ кодируется 16 битами), — 4 Кб. Определите количество символов в тексте. Для решения задачи используйте электронную таблицу, вычисления выполните с помощью формул. Результат сохраните в файле с именем *Задание 1* в папке *Практическая работа 1* своей структуры папок (*1 балл*).

2. Автоматическое устройство перекодировало текстовое сообщение на русском языке, изначально записанное в 16-битной кодировке Unicode, в 8-битную кодировку Windows-1251, при этом длина двоичного кода сообщения составляла 60 байт. Вычислите длину двоичного кода сообщения до перекодирования. Для решения задачи используйте электронную таблицу, вычисления выполните с помощью формул. Результат сохраните в файле с именем *Задание 2* в папке *Практическая работа 1* своей структуры папок (*2 балла*).

3. Сообщение занимает 3 страницы и имеет длину двоичного кода 7950 байт. Определите, сколько строк на странице, если в каждой строке 25 символов и использована кодировка Unicode. Для решения задачи используйте электронную таблицу, вычисления выполните с помощью формул. Результат сохраните в файле с именем *Задание 3* в папке *Практическая работа 1* своей структуры папок (*2 балла*).

4. Представьте текст «Любіть Україну, як сонце, любіть!» в виде кода, используя таблицу кодировки международного стандарта Unicode. Таблицу кодировки найдите в Интернете. Результат представьте в виде текстового документа и сохраните в файле *Задание 4* в папке *Практическая работа 1* своей структуры папок (*3 балла*).

5. Сообщение, набранное на компьютере с использованием кодировки Windows-1251, содержит 1536 символов. Вычислите длину двоичного кода сообщения в килобайтах (*2 балла*).

6. Длина двоичного кода сообщения составляет 0,3 Мб. Выразите это значение в килобайтах (*1 балл*).

7. Найдите ориентировочную длину двоичного кода заданий практической работы. Представьте полученное значение в разных единицах измерения (*3 балла*).

РАЗДЕЛ 2

АППАРАТНО-ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

3. АППАРАТНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

ВСПОМНИТЕ:

- составляющие современного компьютера;
- какие устройства используются для ввода, вывода, хранения данных;
- виды современных персональных компьютеров.

ВЫ УЗНАЕТЕ:

- как технический прогресс повлиял на развитие вычислительной техники;
- какова архитектура современных компьютеров;
- может ли компьютер работать без процессора;
- какие устройства компьютера находятся внутри системного блока;
- что и как можно хранить в памяти компьютера;
- каковы особенности устройств ввода и вывода данных;
- какие устройства относятся к мультимедийному оборудованию;
- как выбрать компьютер для работы.

ИЗУЧАЕМ

1. Как технический прогресс повлиял на развитие вычислительной техники?

Компьютерная техника используется для вычислений и обработки информационных объектов разного типа. Первыми приспособлениями для счёта были, вероятно, зарубки, узелки, счётные палочки. Развиваясь, эти устройства становились сложнее, и со временем появились новые: абак (счёты), логарифмическая линейка, механический арифмометр, электронный компьютер. Средства для вычислений постоянно изменялись и прошли несколько этапов развития (рис. 3.1).

Вычислительная техника постепенно стала использоваться не только для вычислений, но и для решения других задач, например, для автоматизации различных процессов, использования электронных средств связи, контроля оборудования, выполнения офисных задач, компьютерных игр, обучения и т. п. Работники каждой отрасли, в свою очередь, предъявляли дополнительные требования к постоянно развивающемуся компьютерному оборудованию.

Компьютерная техника развивается постоянно в нескольких направлениях. Во-первых, изменяются либо используются новые основные элементы, из которых изготовляют компьютер, — меняется элементная база компьютеров. Во-вторых, создаётся новое программное обеспечение. Кроме того, совершенствуются устройства ввода и вывода данных, организация и взаимосвязь его отдельных составляющих.

Рис. 3.1

Началом эры компьютеров считают 1945–1946 гг., когда американские учёные Проспер Эккерт и Джон Моучли сконструировали в Пенсильванском университете (США) первую электронную вычислительную машину (ЭВМ) «ENIAC» (от англ. — *Electronic Numerical Integrator and Calculator*). В ENIAC было 1800 электронных ламп и 150 000 электромеханических реле. Разумеется, эта машина была очень громоздкой, сложной в управлении (чтобы изменить программу, необходимо было перепаивать схему), ненадёжной в работе, имела ряд других недостатков. ENIAC относят к первому поколению электронных вычислительных машин.

Под **поколениями ЭВМ** понимают все типы и модели электронных вычислительных машин, разработанные разными конструкторскими коллективами, но построенные по одним и тем же научным и техническим принципам. Каждое следующее поколение определяется новыми электронными элементами, технология изготовления которых является принципиально иной, вычислительными возможностями, быстродействием и другими свойствами.

Поколение ЭВМ (англ. — *computer generation*) — один из классов в классификации вычислительных систем по степени развития аппаратных и программных средств.

Обычно выделяют четыре поколения электронной вычислительной техники (табл. 3.1).

Начиная с третьего поколения компьютеры получили широкое распространение.

Таблица 3.1

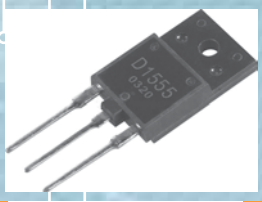
Поколения ЭВМ	Даты	Электронные элементы	Быстродействие
I	1950–1960	Электровакуумные лампы (ENIAC, МЕСМ)	10–20 тыс. оп./с
II	1960–1965	Транзисторы	100–500 тыс. оп./с
III	1965–1970	Интегральные схемы	1 млн оп./с
IV	с 1970	Микропроцессоры	сотни млн оп./с

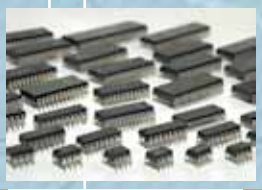
В нашей стране проектирование ЭВМ началось также в 1940-х годах. В 1951 г. в Киеве под руководством профессора Сергея Алексеевича Лебедева (1902–1974) была введена в эксплуатацию ЭВМ, получившая название «МЭСМ» (*Малая электронно-счётная машина*).

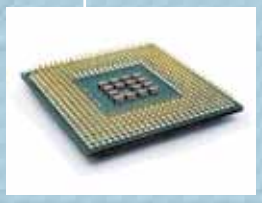
Значительный вклад в развитие отечественной компьютерной техники сделал Виктор Михайлович Глушков (1923–1982) — основатель научной школы кибернетики, автор фундаментальных трудов по кибернетике, искусственному интеллекту, теории цифровых автоматов, вопросам применения кибернетических методов в экономике.

С 1950-х годов вычислительная техника стала бурно развиваться и за рубежом, и в Украине.

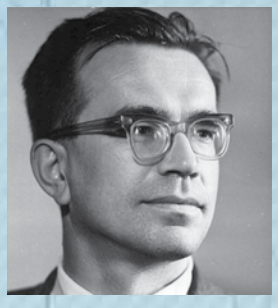

Электровакуумная лампа


Транзистор


Интегральные схемы


Микропроцессор


С. А. Лебедев


В. М. Глушков

ДЕЙСТВУЕМ

Упражнение 1. История развития вычислительной техники.
Задание. Дополните таблицу в файле *Поколения компьютеров*, содержащемся в папке *Обеспечение компьютера*, изображениями компьютеров каждого из поколений и их элементной базы, сведениями о выдающихся людях, имеющих отношение к развитию каждого из поколений ЭВМ.

1. Создайте папку *Компьютеры и программы* в своей структуре папок.
2. В папке *Обеспечение компьютера* откройте файл *Музеи компьютерной техники* и выберите одну из приведённых ссылок. Найдите изображения компьютеров каждого из поколений, их элементной базы и сведения о выдающихся людях, имеющих отношение к развитию каждого из поколений.
3. В папке *Обеспечение компьютера* откройте файл *Поколения компьютеров*. Добавьте в ячейки таблицы найденные сведения.
4. Сохраните файл с именем *Поколения компьютеров_Фамилия* в папке *Компьютеры и программы* своей структуры папок.

2. Какова архитектура современных компьютеров?

Современный стационарный компьютер может иметь следующие составляющие: системный блок, монитор, клавиатуру и мышь, акустическую систему, принтер, сканер и т. п. Сегодня используют и другие виды компьютеров: ноутбуки и нетбуки, планшетные и карманные. В таких компьютерах системный блок, монитор и часто другие устройства объединены в одно целое. Несмотря на то, что области применения современных компьютеров практически не ограничены, в основу их работы положены единые принципы работы. Они определяют общую структуру, без учёта особенностей тех или иных моделей, и отображают основные связи между устройствами компьютера, потоки данных, циркулирующих между ними, и принципы их обработки. Унификация архитектуры ПК обеспечивает их совместимость с точки зрения пользователя.

Архитектура ПК — принципы работы и взаимодействия основных устройств компьютера: процессора, внутренней и внешней памяти, устройств ввода и вывода данных.

Джон фон Нейман

В основу большинства моделей современных компьютеров положена архитектура, предложенная американским математиком Джоном фон Нейманом, — описание логической организации ЭВМ (рис. 3.2).

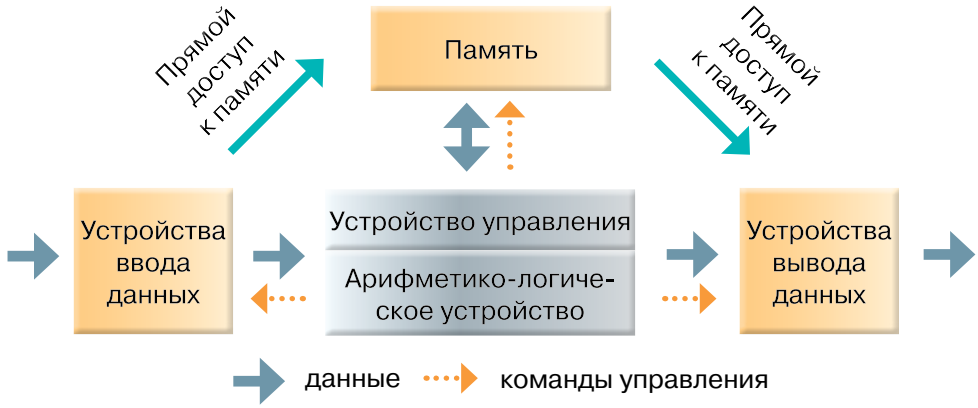


Рис. 3.2

Основные составляющие фон-неймановской машины таковы: устройство управления, арифметико-логическое устройство, память, устройства ввода и вывода данных.

Принципы работы компьютера по фон Нейману

1. С помощью устройств ввода данные и программы для их обработки попадают в память компьютера.
2. Из памяти компьютера данные отправляются в процессор.
3. Арифметико-логическое устройство выполняет обработку данных.
4. Устройство управления обеспечивает выполнение процессов обработки данных, их хранения и передачи.
5. Устройства вывода данных представляют результаты обработки данных в виде, удобном для пользователя.

3. Может ли компьютер работать без процессора?

Процессор называют электронным «мозгом» компьютера. Он предназначен для автоматической обработки и преобразования данных по заранее введённым программам и управления работой всех устройств компьютера. От его вычислительной мощности в большей степени и зависит производительность компьютера.

Процессор — это микросхема (рис. 3.3), которая создаётся на полупроводниковом кристалле (или нескольких кристаллах) путём применения сложной микроэлектронной технологии. Разнообразные операции в процессоре выполняются по специальным командам. Команды для процессора записывают в компьютерной программе.

При работе процессор достаточно сильно нагревается, поэтому на него устанавливают систему охлаждения — вентилятор, который называется **кулером**.

Процессор состоит из:

- арифметико-логического устройства для выполнения арифметических и логических операций с данными;
- устройства управления работой всех составляющих компьютера;
- регистров собственной памяти.

При работе процессор обрабатывает данные. Часть данных интерпретируется как собственно данные, часть данных — как адресные данные, а часть — как команды. Совокупность разнообразных команд, которые может выполнить с данными процессор, образует так называемую систему команд процессора.

Основными характеристиками процессоров являются:

- **тип** — в соответствии с фирмой-производителем различают процессоры Intel (Pentium, Celeron, Core 2 Duo и т. п.), AMD (AMD64, Duron, Athlon и т. п.) и др.;
- **тактовая частота** — определяет количество выполняемых элементарных операций за одну секунду, т. е. быстродействие процессора; тактовая частота современных процессоров измеряется в гигагерцах (ГГц); уже разработаны процессоры с частотой свыше 3 ГГц;
- **разрядность** — максимальная длина двоичного кода, которая может обрабатываться или передаваться процессором; чем выше разрядность, тем мощнее процессор;
- **кэш-память** — это внутренняя память процессора, позволяющая хранить промежуточные данные.

Интересно

Принципы, согласно которым функционирует большинство современных компьютеров, опубликованы в 1946 г. американским математиком Джоном фон Нейманом (1903–1957). Он также описал машину, которая может быть универсальным средством обработки данных.

Рис. 3.3

Интересно

Процессор ещё иногда называют **CPU** — от англ. *Central Processing Unit* — модуль центрального процессора.

Кулер — от англ. *cooler* — охладитель.

Режим работы процессора задаётся специальной микросхемой, которую называют генератором тактовой частоты. Это устройство определяет ритм и скорость работы — на выполнение процессором каждой операции отводится определённое количество тактов.

Объём кэш-памяти современных процессоров колеблется от 256 до 1024 Кб.

Разрядность связана с размерами специальных ячеек памяти, которые содержатся в самом процессоре и называются регистрами. Процессор с регистром 1 байт (8 бит) называют 8-разрядным, 2 байта — 16-разрядным, 4 байта — 32-разрядным. Самые мощные компьютеры сегодня имеют 8-байтовые регистры (64 разряда).

4. Какие устройства компьютера находятся внутри системного блока?

Некоторые устройства расположены внутри системного блока компьютера (рис. 3.4), а другие присоединяются к нему, поэтому они относятся к внешним.

Внутри компьютера расположена **системная плата**, которую ещё называют **материнской**. На ней установлены процессор, внутренняя память компьютера и другие устройства. Процессор соединён с другими устройствами памяти и устройствами для передачи данных и служебных сигналов с помощью набора электронных линий, которые называются **магистралью** (**шиной**). Пользователь может создавать различные конфигурации компьютера, присоединяя к магистрали отдельные модули разных устройств ввода и вывода, памяти и т. п. (рис. 3.5).

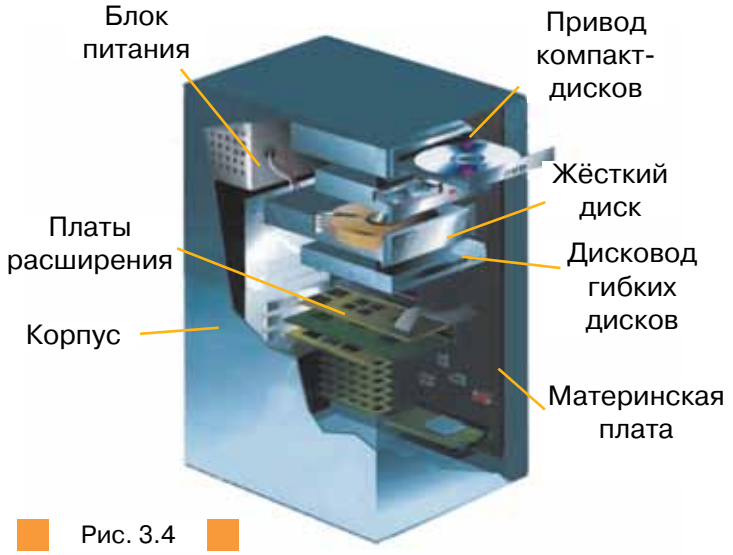
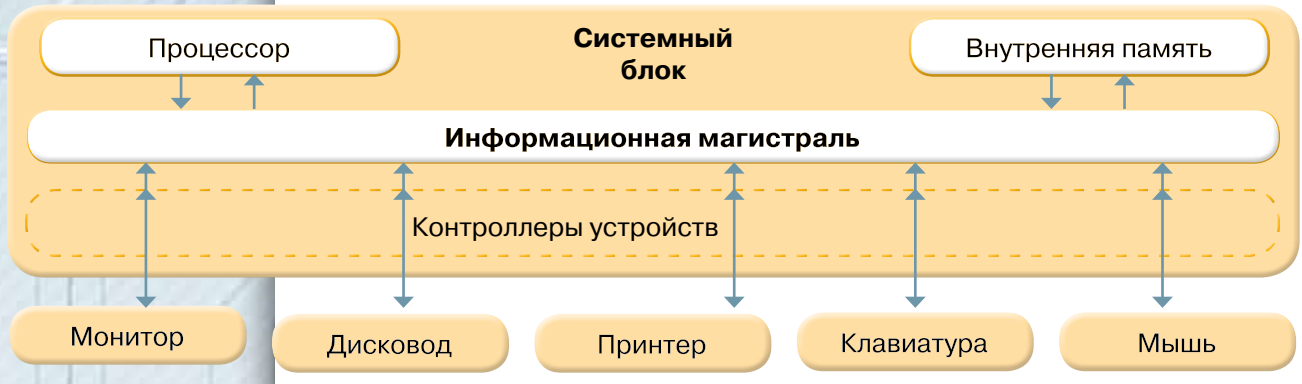


Рис. 3.4

Для магистрали характерна такая организация: по одной группе проводов (шине данных) передаются обрабатываемые данные, по другой (шине адресов) — адреса памяти или внешних устройств, к которым «обращается» процессор. По третьей части магистрали (шине управления) передаются управляющие сигналы (например, проверка готовности устройства к работе, сигнал к началу работы устройства и т. п.).

Рис. 3.5



Пользователь может изменять набор устройств компьютера. Аппаратное подсоединение внешних устройств к магистрали осуществляется через **контро́ллеры** и **адаптеры** — электронные микросхемы, с помощью которых согласуется работа внешних устройств. Они предназначены для преобразования данных, поступающих из процессора, в соответствующие сигналы, с помощью которых осуществляется управление работой устройства. Их разъёмы выведены на заднюю панель системного блока, и с помощью соответствующих кабелей к ним присоединяются внешние устройства.

ДЕЙСТВУЕМ

Упражнение 2. Системный блок.
Задание. В эмуляторе системного блока разместите устройства в правильном месте и нужном порядке.

1. Откройте программу *Системный блок*, находящуюся в папке *Обеспечение компьютера*.
2. Последовательно размещайте блок питания, материнскую плату и другие устройства в места их обычного расположения.
3. Проверьте, получен ли правильный результат.

5. Что и как можно хранить в памяти компьютера?

Память компьютера предназначена для хранения данных и программ. Память бывает внутренняя и внешняя (рис. 3.6).

Устройства внутренней памяти изготовляют в виде микросхем (модулей), которые вставляются в специальные разъёмы на материнской плате.

К внутренней памяти компьютера относятся оперативное запоминающее устройство (ОЗУ), постоянное запоминающее устройство (ПЗУ), полупостоянное программируемое запоминающее устройство (ППЗУ), видеопамять и кэш-память.

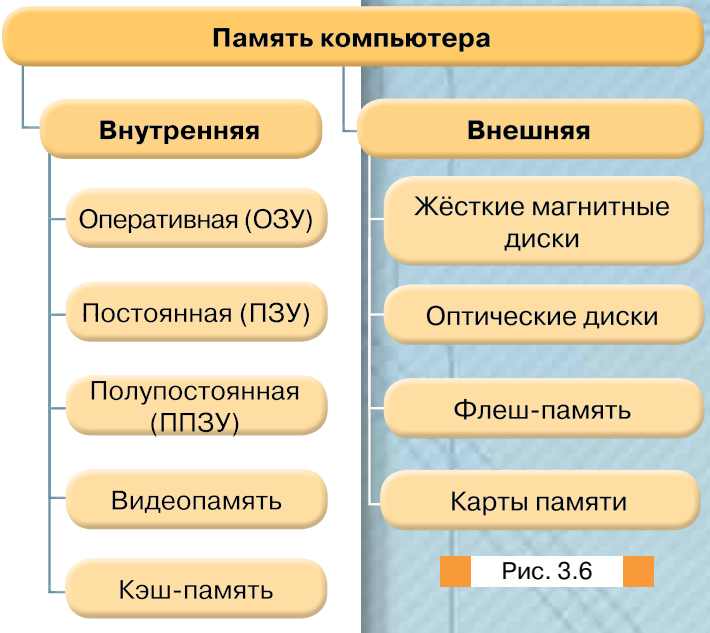


Рис. 3.6

Рассмотрим устройства внутренней памяти.

Оперативное запоминающее устройство — ОЗУ (RAM — от англ. *Random Access Memory* — память со свободным доступом) — быстрая и энергозависимая память (рис. 3.7). Оперативная память предназначена для временного хранения исходных данных, промежуточных и конечных результатов вычислений, программ обработки данных. Это своеобразное рабочее пространство для компьютера. ОЗУ может использоваться как для чтения данных, так и для записи. Данные в ОЗУ хранятся до тех пор, пока на их место не будут записаны новые данные. При отключении электропитания данные в ОЗУ теряются. Объём оперативной памяти современных компьютеров составляет 128, 256, 512, 1024 Мб и даже 4 Гб.

Рис. 3.7

Постоянное запоминающее устройство — ПЗУ (ROM — от англ. *Read Only Memory* — память только для чтения) — быстрая и энергонезависимая память

Рис. 3.8

(рис. 3.8). Данные заносятся в неё один раз навсегда (как правило, в заводских условиях) и хранятся постоянно (при включённом и выключенном питании). Постоянная память — микросхема, в которой содержатся программы для управления работой компьютера и программы тестирования основных его составляющих, а также набор программ для управления всеми его устройствами (BIOS — от англ. *Basic Input/Output System* — базовая система ввода-вывода). Постоянная память также расположена на материнской плате.

Данные, которые хранятся в **полупостоянном программируемом запоминающем устройстве** — ППЗУ (память, выполненная по технологии CMOS, — от англ. *Complementary Metal-Oxide Semicondactor* — технология изготовления микросхем), могут быть заменены в специальном режиме работы компьютера — режиме программирования, когда пользователь владеет специальными знаниями и может написать специальные программы для управления вычислительной машиной. К таким данным относятся данные о хранении и изменении конфигурации компьютера, календаря и часов. ППЗУ также называют памятью автономного питания или памятью «на батарейках», поскольку данные хранятся с помощью аккумуляторной батарейки, по своим функциям подобной батарейкам кварцевых часов.

Рис. 3.9

Видеопамять — быстрая оперативная память для хранения кода изображения, отображающегося на экране монитора. Видеопамять (VRAM — от англ. *Video Random Access Memory*) размещена на видеокарте (рис. 3.9). Высокопроизводительные видеокарты применяют для компьютерных игр или для работы с пространственными изображениями.

Чем больше объём видеопамяти компьютера, тем больше возможность отображения на мониторе графиков с высокой разрешающей способностью и большим количеством цветов. Объём видеопамяти современных компьютеров составляет 64, 128, 256 Мб и более.

Кэш-память — это специальный вид памяти или части ОЗУ, где хранятся копии часто используемых данных. Кэш-память обеспечивает быстрый доступ к ним.

Кэш-память современных компьютеров имеет несколько уровней: кэш-память первого уровня объёмом 32 КБ встраивается в процессор, а кэш-память второго уровня объёмом до 2 Мб обычно размещается на материнской плате.

Рис. 3.10

Для длительного хранения данных предназначена **внешняя память**, или носители данных:

- жёсткий магнитный диск (ЖМД, HDD — от англ. *Hard Disk Drive*), или винчестер (рис. 3.10). Как правило, встроен вместе с дисководом в корпус системного блока (он может быть размещён и снаружи);
- лазерные диски (CD-ROM, CD-R, CD-RW или DVD) (рис. 3.11):
  - диски CD-ROM (от англ. *Compact Disk Read Only Memory* — компакт-диски только для чтения) — высоконадёжные носители для хранения данных, долговечные (срок пригодности, при качественном изготовлении, до 50 лет). Диаметр диска может быть как 5,25, так и 3,5 дюйма. Принцип записи и считывания — оптический;
  - диски CD-R (от англ. *Compact Disc Recordable* — компакт-диск для однократной записи) — это разновидность оптического диска, на котором можно записать файлы с помощью записывающего устройства. Записанные данные можно считать с диска с помощью привода CD-ROM или воспроизвести в проигрывателе компакт-дисков (если это музыка). На оптические диски этого вида данные можно дописывать, пока не закончится свободное место. Обычно на диске CD-R можно поместить

Рис. 3.11

650 Мб данных или 74 мин музыки. На CD-R дисках нового поколения можно дополнительно увеличить этот лимит до 737 Мб или 80 мин музыки. Современные CD-R могут иметь объём до 800 Мб;

- диски CD-RW (от англ. *Compact Disc ReWritable* — компакт-диски с возможностью перезаписи) — это другой вид оптических дисков, на которых можно не только записывать данные, но и удалять и перезаписывать их. У них такой же объём, как у CD-R;
- диски DVD (от англ. *Digital Video Disc* — цифровой видеодиск или *Digital Versatile Disc* — цифровой многофункциональный диск) — это разновидность носителя данных, внешне он напоминает диск CD-ROM. Однако на DVD-диске можно записать значительно больше данных. Стандартный объём этих носителей составляет 4,7 Гб, хотя встречаются диски вдвое большего объёма. На DVD-диске можно записать в совершенном качестве полнометражный фильм на нескольких языках;

- флеш-память, или USB-накопители (от англ. *Universal Serial Bus* — универсальная последовательная шина) (рис. 3.12), подсоединяются непосредственно к порту USB на компьютере. Они представляют собой микросхемы и могут хранить до нескольких гигабайтов данных;
- карты памяти, используемые для хранения данных в цифровых фотоаппаратах, смартфонах и других устройствах (рис. 3.13). Для перенесения на компьютер объектов, хранящихся на картах памяти, используют кардридер.

Рис. 3.12

Рис. 3.13

6. Каковы особенности устройств ввода и вывода данных?

Устройства ввода и вывода (рис. 3.14) предназначены для ввода данных в компьютер и вывода результатов их обработки в удобном для пользователя виде.

Рис. 3.14

С основными устройствами ввода данных — клавиатурой и мышью — вы уже знакомы.

Основным устройством вывода данных является **монитор (дисплей)**. Мониторы обладают следующими характеристиками:

- качество отображения цветных изображений, т. е. количество цветов для отображения;

Рис. 3.15

Рис. 3.16, *а*

Рис. 3.16, *б*

Интересно

Обычно ноутбуки не оснащены видеоадаптерами. Они имеют интегрированную графику или графический процессор (GPU), встроенный в компьютер вместо видеоадаптера.

- разрешающая способность, которая определяется количеством точек (пикселей) на экране, используемых для создания изображения. Разрешающая способность определяется как произведение количества пикселей по горизонтали на количество пикселей по вертикали; например, 800×600 или 1024×768. Чем выше разрешающая способность, тем детальнее можно отобразить изображение;
- длина диагонали в дюймах;
- размер зерна (расстояние на экране между двумя точками одинакового цвета);
- максимальная частота обновления изображения, на которую способен монитор (её измеряют в герцах), например, 120 Гц. Чем выше частота обновления, тем больше плавность воспроизведения изображения.

По принципу изготовления различают мониторы на основе электронно-лучевой трубки (рис. 3.15) и жидкокристаллические (рис. 3.16, *а*, *б*).

Самым старым типом являются мониторы на основе электронно-лучевой трубки — CRT (от англ. *Cathode Ray Tube* — катодно-лучевая трубка). CRT-монитор имеет очень малые размеры зерна и незначительное опоздание во времени изменения изображения, он качественно отображает цвета. Однако такие мониторы занимают много места и в наше время активно вытесняются жидкокристаллическими мониторами.

У жидкокристаллических мониторов — LCD (от англ. *Liquid Crystal Display* — жидкокристаллический дисплей) — полностью отсутствует вредное электромагнитное излучение, потребляется меньше электроэнергии, не создаётся эффект мерцания, изображение не искажается. У них плоский (рис. 3.16, *а*), а в новейших моделях — вогнутый (рис. 3.16, *б*) экран, поэтому они занимают меньше места на столе. В жидкокристаллических мониторах применяется технология TFT (от англ. *Thin Film Transistor* — тонкоплёночный транзистор).

Для вывода изображений на монитор используют видеопамять, размещённую на видеокарте (рис. 3.9). Всё, что пользователь видит на экране монитора, содержится в видеопамяти, из которой поступают видеосигналы на монитор. **Видеокарта**, которая называется также **видеоадаптером**, устанавливается внутри компьютера и используется для подключения монитора и передачи на него графических данных.

Параметры видеокарты определяют максимальную разрешающую способность, количество цветов и частоту обновления изображения. Чем больше объём видеопамяти, тем большим количеством пикселей на экране может управлять видеоадаптер, т. е. иметь более высокую разрешающую способность экрана. Чем больший объём видеопамяти используется для управления одним пикселем, тем больше количество воспроизводимых цветов, следовательно, более богатая цветовая палитра монитора.

В современных компьютерах преимущественно используются мониторы с такими характеристиками:

- количество цветов — 256 и более;
- разрешающая способность — 800×600 (1024×768, 1280×1024);
- размер экрана — 15, 17, 19 дюймов.

Для вывода данных на бумагу, плёнку или другой носитель используют **принтеры** (рис. 3.17). Основными характеристиками принтеров являются качество печати (dpi — количество точек на дюйм) и скорость печати (количество страниц в минуту — стр./мин).

Для домашнего применения обычно используют **струйный** принтер, позволяющий печатать как чёрно-белые, так и цветные документы. В струйных принтерах для формирования изображения используются специальные сопла, через которые на бумагу подаются чернила. Тонкие, как волоски, сопла находятся на печатной головке принтера, где установлены резервуары с жидкими чернилами, которые, как микрочастицы, переносятся через сопла на материал носителя. Количество сопел зависит от модели принтера и его производителя. Обычно их бывает от 16 до 64. В некоторых последних моделях количество сопел намного больше: так, например, у головки принтера DeskJet 1600 300 сопел для чёрных чернил и 416 — для цветных. Такие принтеры осуществляют качественную малошумную печать. Но для них нужны дорогие расходные материалы, к тому же чернила могут расплываться в воде.

Матричные принтеры исторически были самыми первыми принтерами и в своё время — самыми распространёнными. В настоящее время их доля составляет менее 10 %. Невысокое качество чёрно-белой печати до 300 dpi, возможность печати нескольких копий одновременно под копировальную бумагу и достаточная дешевизна печати обеспечивают их использование в банках и для печати чеков. Работает матричный принтер шумно и с низкой скоростью. Способ печати матричных принтеров похож на способ печати обычной печатной машинки, но матричный принтер сам формирует любое изображение символов с помощью игл, ударяющих по покрасочной ленте.

В офисах, где необходимо печатать большое количество документов с небольшим шумом, используют **лазерные** принтеры. Качество печати на таких принтерах достигает до 1200 dpi, а скорость — 25–50 стр./мин. Расходные материалы для них относительно недорогие, хотя существуют определённые требования к качеству бумаги. Как и копировальные аппараты, лазерные принтеры используют фотобарабан, на который лазерным лучом наносится изображение. Электрически заряженный порошок — тонер — притягивается к месту, где отработал лазерный луч. Тонер переносится на бумагу и закрепляется на ней горячим спеканием с помощью разогретого валика.

Рис. 3.17

К **специальным** принтерам относятся фотопринтеры, 3D-принтеры и другие устройства, используемые для печати профессиональных изображений или объектов; в быту, как правило, их не используют.

3D-принтер — устройство, работающее методом послойного создания физического объекта по цифровой 3D-модели (объёмной). Существует несколько технологий 3D-печати от формирования объекта из порошка до постепенного его наслоения из специального полимера. Такие принтеры используются как в мелкосерийной инженерии, так и в производстве сложных систем, даже биологических. Эти технологии в последнее время очень быстро развиваются, постепенно становятся дешевле и доступнее для пользователей.

7. Какие устройства относятся к мультимедийному оборудованию?

Использовать мультимедийные программы и обрабатывать мультимедийные данные в компьютере можно с помощью специального оборудования.

Минимальный набор мультимедийного оборудования состоит из **звуковой карты** (платы, присоединяющейся к материнской плате), к которой через соответствующую панель системного блока подсоединяется **акустическая система (колонки)**, и **накопителя для оптических дисков**. Звук, который слышит пользователь компьютера, — результат работы двух взаимосвязанных компонентов: звуковой карты (рис. 3.18) и акустической системы. Их выбор зависит от требуемого качества звука и от области использования ПК (игры, домашний мультимедийный центр, домашний кинотеатр, для просмотра DVD-видео и т. п.). Однако качество воспроизведения звука зависит не только от устройств, но и от программного обеспечения.

Рис. 3.18

Важными характеристиками акустических систем являются:

- диапазон частот. Обычно в пределах от 20 Гц до 20 кГц. Это достаточно широкий диапазон, и для его воспроизведения необходимо несколько динамиков;
- количество динамиков. Каждый динамик воспроизводит свой узкий диапазон частот;
- мощность. Составляет от 2 до 180 Вт.

Вместо колонок можно использовать наушники (рис. 3.19). Разъём для наушников есть во многих колонках или на задней панели системного блока. Популярными становятся колонки и наушники с пространственным звуком.

Рис. 3.19

К более широкому комплекту мультимедиа-системы относятся **микрофон**, **видеокамера**, **видеопроигрыватель**, **цифровая фотокамера**.

С помощью микрофона можно записать звуковой фрагмент и сохранить его как файл. Обычно микрофон используют для общения в Интернете с помощью средств IP-телефонии.

Микрофон не является базовым мультимедийным устройством, поэтому, прежде чем его купить, необходимо проверить совместимость микрофона с установленной звуковой платой. Микрофон подсоединяется к соответствующему разъёму звуковой платы или к линейному входу. У большинства микрофонов есть выключатель для выключения исходного сигнала (вместо отсоединения от звуковой платы).

Физическое подсоединение перечисленных устройств должно сопровождаться установкой соответствующих программ — **драйверов** (от англ. *to drive* — управлять, вести), управляющих работой внешних устройств компьютера. Как правило, соответствующие драйверы записаны на CD-дисках и входят в комплект при продаже устройства, поскольку у каждого типа внешнего устройства есть свой индивидуальный драйвер. Создавая мультимедийный центр, следует помнить, что эффективная работа на компьютере с видео и графикой требует определённых характеристик процессора, оперативной памяти, жёсткого диска.

8. Как выбрать компьютер для работы?

Если пользователь не собирается работать с профессиональными графическими программами или обрабатывать большое количество данных, то ему не нужна самая дорогая модель компьютера и можно выбрать более «медленный» процессор.

Нужно также учитывать, что скорость обработки данных зависит от комплектующих компьютера, т. е. одни могут улучшить производительность компьютера (например, память RAM, видеокарта), а другие — повлиять на комфорт

работы пользователя (объём жёсткого диска, размер и яркость монитора, качество звуковой карты). Основные характеристики персонального компьютера зависят от характеристик его комплектующих.

Но в любом случае для выбора компьютера необходимо знать о его назначении, функциональности и особенностях работы основных комплектующих.

ДЕЙСТВУЕМ

Упражнение 3. Конфигурация компьютера.

Задание. Определите характеристики компьютера, за которым вы работаете.

1. Щёлкните правой кнопкой мыши на значке *Компьютер* на *Рабочем столе* и выберите команду *Свойства*.
2. Просмотрите и проанализируйте следующие характеристики компьютера в области *Просмотр основных сведений о вашем компьютере*: версия операционной системы, характеристики процессора, объём оперативной памяти, тип системы.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 3* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Приведите три примера разных видов компьютеров, с которыми вы сталкиваетесь в повседневной жизни. Определите их общие характеристики. Результаты представьте в виде диаграммы Венна. Сравните полученный результат с диаграммой другой пары.
2. Обсудите в парах: для чего предназначены отдельные устройства компьютера. Один ученик называет устройство, относящееся к компьютеру, а другой — его назначение; потом поменяйтесь ролями и следите, чтобы названия устройств не повторялись.
3. Упорядочьте носители данных по степени надёжности, стоимости, скорости доступа, новизне и т. п. Результаты обсудите в парах.
4. Без каких устройств невозможно слушать музыку на компьютере? При каких условиях можно прослушать на компьютере концерт классической музыки, записанный на CD?
5. При каких условиях на компьютере можно посмотреть видеофильм? А поиграть в компьютерную игру?
6. Только ли в компьютерах есть процессоры? Назовите другие технические устройства, в которых может быть процессор. Где используются эти устройства? Какие функции выполняют процессоры в приведённых вами примерах? Похожи ли их функции на функции процессора ПК? Могут ли существовать эти устройства без процессоров? Изменится ли при этом их назначение и эффективность?
7. Найдите в справочной литературе данные о современных процессорах и перспективах их развития. Какова основная технологическая проблема создания современных процессоров? Изготовляет ли Украина процессоры для компьютеров? Какие корпорации являются известными производителями процессоров? Обсудите, как лучше представить полученные сведения, чтобы поделиться ими со своими одноклассниками. Составьте план выступления.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Создайте модель компьютера, воспользовавшись набором устройств в папке *Устройства* папки *Обеспечение компьютеров,* для:
   а) обучения и исследований;
   б) игр и развлечений;
   в) ведения деловой документации.
2. Найдите в Интернете видеозаписи и изображения различных приспособлений для счёта, которые использовались в древности.
3. Найдите в Интернете информацию о том, каким годам соответствуют этапы развития вычислительных средств, изображённые на рисунке 3.1.
4. Спланируйте и составьте презентацию *Украина в истории вычислительной техники*.
5. Составьте карту знаний классификации устройств компьютера. Предусмотрите в ней стандартные и дополнительные устройства ввода, вывода, памяти и процессор. Для справки воспользуйтесь файлом *Дополнительные устройства* в папке *Обеспечение компьютеров*.
6. Определите конфигурацию домашнего компьютера. Воспользуйтесь ценами в интернет-магазине и определите стоимость подобного компьютера сегодня. Сделайте вывод.

4. ПРАКТИЧЕСКАЯ РАБОТА 2

КОНФИГУРАЦИЯ КОМПЬЮТЕРА ДЛЯ ПОЛЬЗОВАТЕЛЯ

ВСПОМНИТЕ

- Назначение составляющих компьютера;
- стандартную архитектуру персонального компьютера;
- основные характеристики запоминающих устройств, процессоров, мониторов, видеоадаптеров, принтеров.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 2.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Конфигурация компьютера (*12 баллов*)

Предложите, как, затратив минимум ресурсов, оборудовать компьютером рабочее место для:

- ученика на олимпиаде по компьютерной графике;
- сотрудника бюджетного учреждения, работающего в бухгалтерии;
- ученика, который будет использовать мультимедиа и игры;
- студента;
- офисного работника, работающего с большим количеством документов.

Назначение компьютера для оборудования выберите из предложенного списка. Приведите два примера конфигурации. Укажите общую стоимость оборудования. Результаты работы представьте в презентации.

Задание 2. Назначение компьютера (*5 баллов*)

По конфигурации компьютера определите, какие задания можно выполнять на нём. Для каких пользователей вы бы его рекомендовали? Запишите свои выводы в текст электронного письма и отправьте его на учебный электронный ящик, адрес которого укажет учитель.

Система

Оценка	5,5 Индекс производительности Windows
Процесор	Intel(R) Core(TM)2 Duo CPU T9400 @ 2.53GHz 2.53GHz
Установленная память (ОЗУ):	2,00 Гб
Тип системы	64-разрядная операционная система
Перо и сенсорный ввод:	Перо и сенсорный ввод недоступны для этого экрана

Задание 3. Устройства ввода и вывода (*12 баллов*)

Создайте текстовый документ с рекомендациями по приобретению устройств ввода и вывода данных. В документе предусмотрите названия устройств, 2–3 примера моделей и их изображения, цену и ссылку на интернет-магазин для возможного приобретения, советы по задачам, которые можно решать с помощью этих устройств. Необходимые материалы найдите в Интернете.

5. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА

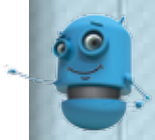
ВСПОМНИТЕ:

- что называется программой; где отображаются названия программ, установленных на компьютере;
- как запустить программу на выполнение;
- какие программы называются операционной системой;
- когда следует соблюдать авторские права;
- что такое интерфейс операционной системы и как он реализуется в операционной системе *Windows* 7.

ВЫ УЗНАЕТЕ:

- какое программное обеспечение требуется для работы компьютера;
- каковы правила использования программного обеспечения;
- какие бывают виды лицензий на программное обеспечение;
- для чего нужна операционная система, каковы её функции;
- как классифицируются операционные системы;
- какие программы относятся к служебным программным средствам;
- когда программное обеспечение требует инсталляции и деинсталляции;
- в каких случаях говорят о проблемах совместимости программного обеспечения;
- что такое форматирование носителей данных и с помощью какого программного обеспечения оно осуществляется;
- какие функции выполняют программы-архиваторы;
- в чём заключаются основные методы сжатия данных.

ИЗУЧАЕМ



1. Какое программное обеспечение требуется для работы компьютера?

Вы уже знаете, что без программ компьютер не сможет работать, а будет лишь набором электронных устройств.

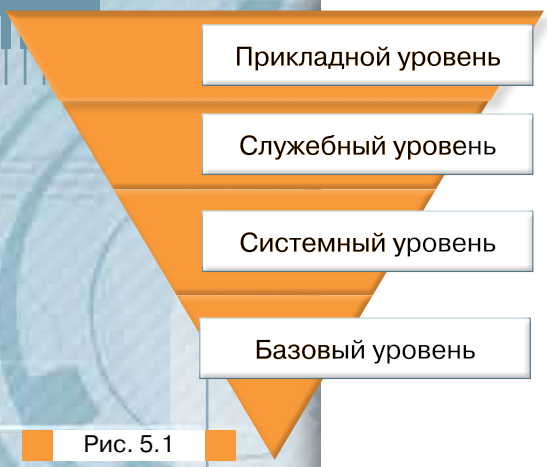


Рис. 5.1

Для решения задач на компьютере нужно, чтобы каждая программа была настроена и имела соответствующую документацию. Поэтому при работе на компьютере часто употребляют термин **программное обеспечение (software)**, подразумевающий совокупность программ и правил, а также документации по работе компьютера для обработки данных.

Различают программы разного уровня, каждый из которых имеет соответствующее назначение. Схематически структура программного обеспечения показана на рисунке 5.1.

Программы **базового уровня** хранятся в специальных микросхемах постоянного запоминающего устройства и создают базовую систему ввода-вывода — BIOS. Программы и данные записываются в ПЗУ на этапе производства и не могут быть изменены в процессе эксплуатации. Программы этого уровня обеспечивают взаимодействие с базовыми аппаратными средствами.

Программы **системного уровня** обеспечивают взаимодействие других программ компьютера с программами базового уровня и непосредственно с аппаратным обеспечением. При подключении к компьютеру нового оборудования на системном уровне должна быть установлена программа, обеспечивающая взаимосвязь остальных программ с этим устройством. Программы, обеспечивающие взаимодействие с конкретными устройствами, называются **драйверами**. Другой класс программ системного уровня отвечает за взаимодействие с пользователем. Благодаря таким программам можно вводить данные в компьютер, управлять его работой и получать результат в удобном для пользователя виде. Это средства обеспечения интерфейса пользователя, от них зависит удобство и производительность работы с компьютером.

Программы **служебного уровня** взаимодействуют как с программами базового уровня, так и с программами системного уровня. Назначение служебных программ (утилит) заключается в автоматизации проверки и настройки компьютера, а также в улучшении функций системных программ, повышении эффективности работы компьютера и расширении возможностей его использования. К таким программам относятся программы для работы с архивами данных (например, *7-zip*, *WinRar*), оптимизации размещения данных на диске (например, *defrag*), антивирусные программы (например, *Intel Security-McAfee*, *Symantec Norton™ Security*, *Zillya! Internet Security*), программы тестирования компьютера (например, *ScanDisk*) и т. п.

Программное обеспечение **прикладного уровня** представляет собой комплекс прикладных программ, с помощью которых выполняются конкретные задачи (от производственных до творческих, развлекательных и учебных). Между прикладным и системным программным обеспечением существует тесная взаимосвязь. Прикладные программы предназначены для компьютерной поддержки выполнения прикладных задач. Различают прикладные программы общего и профессионального (специального) назначения (рис. 5.2).

К прикладному программному обеспечению **общего назначения** относятся: текстовые и графические редакторы, процессоры, программы создания мультимедийных презентаций, табличные процессоры, средства поддержки коммуникаций и т. п.

Прикладное программное обеспечение общего назначения включает также программы для компьютерной поддержки изучения различных учебных предметов, иностранных языков, виртуальные физические и химические

лаборатории, программы для электронного перевода с иностранных языков, быстрого набора текста на клавиатуре и т. п. Удобны в использовании электронные справочники и энциклопедии. Главное их преимущество перед бумажными аналогами — компактность и удобство в поиске данных.

Прикладное программное обеспечение **профессионального (специального) назначения** используется для решения узкоспециализированных задач. В таких программах учитывается специфика конкретных задач, и составлены они на основе специальных методов представления и обработки данных, используемых в конкретных отраслях деятельности людей. К программам профессионального назначения относятся программы для проведения математических вычислений (например, *Mathlab*, *MathCad*), системы автоматизированного проектирования (например, *AutoCad*), программы для проведения бухгалтерских операций (например, *1С-бухгалтерия*), редакторы трёхмерной графики и анимации (например, *3D MAX Studio*) и т. п.

Для разработки как прикладного, так и системного программного обеспечения существуют специальные программы — **инструментальное программное обеспечение**. Такими инструментальными средствами являются **системы программирования**, предназначенные для создания и обработки программ, записанных на одном или нескольких языках программирования (например, *C*, *C#*, *C++*, *Pyton, Visual Basic*, *Delphi*). Кроме того, существуют системы, поддерживающие несколько языков программирования, например, *Microsoft Visual Studio.NET*.

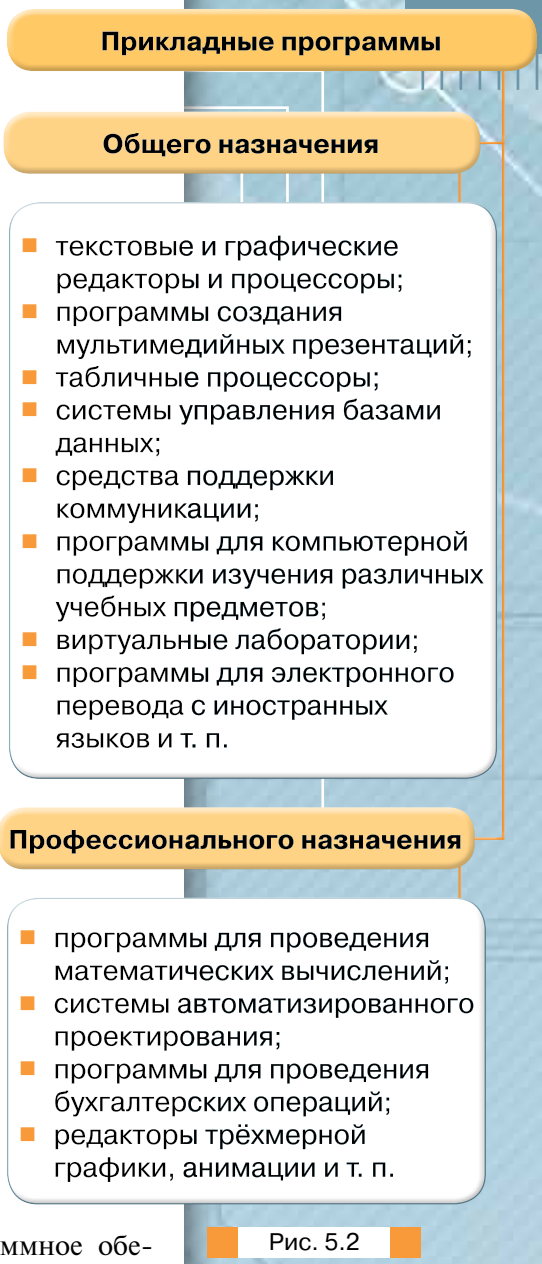


Рис. 5.2

2. Каковы правила использования программного обеспечения?

Компьютерные программы создают программисты. Разработка нового программного обеспечения (ПО) — трудоёмкий и длительный процесс, требующий глубоких знаний и определённых навыков, в первую очередь, в области математики и информатики. Любые программы имеют цену и владельца, т. е. кому-то принадлежат.

Большинство программных продуктов являются коммерческими, т. е. предусматривают плату за их использование. Например, на платной основе распространяется такое программное обеспечение, как операционная система *Windows*, пакет прикладных офисных программ *Microsoft Office*, антивирусное программное обеспечение, программы для компьютерной поддержки обучения, компьютерные игры.

Любое программное обеспечение распространяется на основе лицензионных соглашений, а не только продаётся и покупается. Программы, распространяемые бесплатно, также предусматривают ознакомление с лицензионным соглашением и подтверждение пользователем соблюдения правил их использования, которые определены таким соглашением.

Лицензия на программное обеспечение — правовой документ, определяющий правила использования и распространения программного обеспечения.

Рис. 5.3

Программное обеспечение — это объект интеллектуальной собственности, все права на него принадлежат разработчику.

Это право защищается Законом Украины «Об авторском праве и смежных правах». Согласно закону при продаже программного обеспечения разработчик не передаёт конечному пользователю свои права на программу, а лишь позволяет использовать (лицензирует) эту программу. На упаковке программного продукта, как правило, фиксируется краткое лицензионное соглашение, определяющее основные права и обязанности производителя и владельца полученного программного продукта.

К сожалению, не все продаваемые программные продукты легальны. С юридической точки зрения к пиратскому программному обеспечению относятся все компьютерные программы, которые распространяются, устанавливаются на компьютеры и используются с нарушением условий их лицензионного соглашения. Например, самостоятельно созданные копии лицензионной программы перестают быть легальными, а подобные действия являются нарушением авторских прав и влекут правовую ответственность.

Лицензионное программное обеспечение преимущественно записывается на отдельный компакт-диск и поставляется вместе с сопроводительной документацией в цветной картонной коробке или в специальной упаковке (рис. 5.3). Доступ к лицензионному программному обеспечению можно получить также на сайте производителя, указав персональный пароль лицензии.

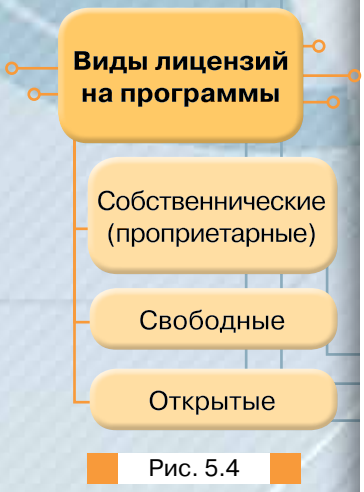


Рис. 5.4

3. Какие бывают виды лицензий на программное обеспечение?

Существуют различные виды лицензий на программы. Основные из них (рис. 5.4):

- собственнические (проприетарные);
- свободные;
- открытые.

Они существенно отличаются в праве конечного пользователя на использование программы.

Собственническая, или **проприетарная**, **лицензия** (от англ. *proprietary* — собственнический) предусматривает, что разработчик ПО разрешает пользователю использовать одну или несколько копий программы, но при этом сам остаётся собственником всех этих копий. Таким образом, практически все права на ПО принадлежат разработчику, а пользователь получает лишь очень ограниченный набор прав. Для проприетарных лицензий характерно большое количество условий, запрещающих определённые варианты использования ПО, даже те, которые без этого запрета были бы разрешены законом об авторском праве. Примером проприетарной лицензии может быть лицензия на операционную систему *Microsoft Windows*, содержащая большой список запрещённых вариантов использования.

Если используется проприетарная лицензия, конечный пользователь обязан её принять, потому что по закону владельцем ПО является не пользователь, а разработчик программы. В случае отказа принять лицензию пользователь вообще не может работать с программой.

Свободные и **открытые** лицензии не оставляют права на конкретную копию программы её разработчику, а передают важнейшие из них конечному пользователю, который и становится владельцем. В итоге пользователь получает важные права, которые закон об авторском праве обычно предоставляет лишь владельцу копии. Однако все авторские права на ПО, как и раньше, остаются у разработчика.

Лицензия *Freeware* (бесплатное ПО) не нуждается в выплатах собственнику, не имеет ограничений относительно функциональности и времени работы. Однако такое ПО можно распространять без текста программы, и могут существовать ограничения в её коммерческом использовании или модификации.

Лицензия *Free software* (свободное ПО) предоставляет пользователям максимальные права: использование, распространение, модификацию. Для этого типа ПО создаются специальные лицензии для урегулирования прав и обязанностей авторов и пользователей.

Отличие между бесплатным и свободным ПО заключается в том, что согласно лицензии *Freeware* пользователи не имеют права распространять программу, дарить, вносить изменения и т. п., а в соответствии с *Free software* — это разрешается. Часто, хотя и не всегда, они отличаются ещё и тем, что свободное ПО поставляют вместе с текстом программы.

4. Для чего нужна операционная система, каковы её функции?

Основой системного программного обеспечения является операционная система (например, *Windows*, *Linux*).

Операционная система (ОС) — это программный комплекс, обеспечивающий:

- управление ресурсами — согласованную работу всех аппаратных средств компьютера;
- управление процессами — выполнение всех программ и их взаимодействие с устройствами компьютера и данными;
- взаимодействие (обмен сведениями и данными) между пользователем и компьютером.

Операционная система начинает работать сразу после включения компьютера. Определённая её часть — BIOS — размещена на микросхеме постоянной памяти. Эта часть содержит программы, которые после включения компьютера автоматически тестируют все его устройства и в случае их исправной работы загружают в оперативную память часть операционной системы — программу-загрузчик. Далее она загружает в оперативную память компьютера необходимые для последующей работы модули операционной системы. После завершения загрузки ОС управление переходит к командному процессору — части ОС, обеспечивающей выполнение команд пользователя. Пока компьютер будет работать, некоторая часть операционной системы будет всегда оставаться в ОЗУ. Эту часть ОС называют резидентной. При необходимости в оперативную память будут подгружены другие модули ОС.

В состав современных операционных систем входят следующие основные компоненты (рис. 5.5):

- **ядро** — центральная часть ОС, обеспечивающая прикладным программам координированный доступ к ресурсам компьютера (времени, которое затрачивает процессор на обработку отдельных заданий, оперативной памяти, внешним устройствам ввода и вывода данных), переводя их команды с языка прикладных программ на язык двоичных кодов для последующей обработки компьютером;
- **драйверы** — программы для перевода команд компьютера на язык определённого устройства (принтера, сканера, звуковой или видеокарты и т. п.) и наоборот;
- **утилиты** — вспомогательные программы, предназначенные для обслуживания дисков, проверки компьютера, настройки параметров работы;

Интересно

Текст программы (исходный код, программный код, англ. *source code*) — набор инструкций или команд, написанных на компьютерном языке программирования и в форме, которую может прочитать и модифицировать человек. Текст программы предоставляет возможность программисту изучать и изменять работу программы в зависимости от потребностей пользователя.

Рис. 5.5

- **интерфейс** — правила взаимодействия операционной системы и пользователя, определяющие удобство работы.

К основным функциям операционной системы относятся:

- создание среды выполнения и взаимодействия прикладных программ;
- распределение аппаратных ресурсов компьютера между прикладными программами;
- предоставление прикладным программам средств для эффективного использования устройств и выполнения основных операций ввода и вывода данных;
- хранение данных устройствами памяти;
- предоставление интерфейса, с помощью которого пользователи будут управлять выполнением прикладных программ и содержимым устройств памяти;
- обеспечение взаимодействия компьютеров в сетях.

В последнее время операционные системы на компьютеры устанавливают производители или компании, занимающиеся комплектацией, продажей и обслуживанием компьютеров. Однако пользователи при необходимости могут самостоятельно установить или заменить операционную систему на своём компьютере. Для этого необходимо приобрести соответствующий пакет программ, проверить наличие лицензии, поскольку компьютерные программы защищены законом об авторском праве, и установить по определённым правилам операционную систему на компьютер.

5. Как классифицируются операционные системы?

Операционные системы (ОС) можно классифицировать по таким признакам:

- целевое назначение для:
  - больших универсальных высокопродуктивных ЭВМ (мэйнфреймов);
  - ПК;
  - мобильных устройств;
  - встроенных систем;
- количество пользователей, одновременно работающих с системой: локальные — **однопользовательские** и сетевые — **многопользовательские**;
- количество задач, которые одновременно может решать пользователь с их помощью: **однозадачные** и **многозадачные**;
- интерфейс пользователя — основной способ взаимодействия пользователя с ОС:
  - интерфейс командной строки — управление с помощью введённых с клавиатуры команд;
  - графический интерфейс — выбор из меню или указание на графические изображения (WIMP-интерфейс, от англ. *Window*, *Icon*, *Menu*, *Pointer* — окно, образ, меню, указатель);
  - SILK-интерфейс — ввод команд голосом (от англ. *Speech*, *Image*, *Language*, *Knowledge* — речь, образ, язык, знание);
  - жестовый интерфейс — управление с помощью сенсорного экрана, джойстика и т. п.;
- ресурсы, минимально необходимые для её работы: минимальный объём оперативной и дисковой памяти, тип процессора;
- открытость: возможность для пользователя, владеющего языками программирования, вносить нужные изменения в её отдельные модули;
- число разрядов, одновременно обрабатывающих данные в процессоре:
  - 16-разрядные;
  - 32-разрядные;
  - 64-разрядные.

Рис. 5.6

Наиболее известные операционные системы: *MS Windows*, *GNU/Linux*, *UNIX*, *OS/2*, *MacOS*, *iOS*, *Android* (рис. 5.6). Одной из первых известных операционных систем была *MS-DOS*, которую в прошлом устанавливали на большинство компьютеров, однако со временем её заменили на ОС с графическим интерфейсом.

Например, операционные системы *Windows* и *Linux* обеспечивают работу нескольких пользователей одновременно (сетевые), а *MS-DOS* — однопользовательская операционная система; операционные системы *MS-DOS* и *UNIX* предусматривают ввод пользователем всех команд с клавиатуры, а при работе с операционной системой *Windows* пользователю, чтобы начать выполнение операции, достаточно выбрать на экране компьютера графические объекты и меню. Операционная система *Linux*, в отличие от ОС *Windows*, имеет открытый код.

Интересно

На одном компьютере можно установить несколько операционных систем. Для этого используют **виртуальную машину** — специальную программу, обеспечивающую полную эмуляцию физической машины или среды.

6. Какие программы относятся к служебным программным средствам?

К служебным программам относятся такие группы программ (рис. 5.7).

Диспетчеры файлов (файловые менеджеры). С их помощью выполняется большинство операций по обслуживанию файловой структуры: копирование, перемещение, переименование файлов, создание папок, удаление объектов, поиск файлов и навигация в файловой структуре. Базовые программные средства входят в состав программ системного уровня и устанавливаются вместе с операционной системой.

Архиваторы (средства сжатия данных). Предназначены для создания архивов. Файлы архивов имеют повышенную плотность записи данных, за счёт чего эффективнее используются носители данных.

Средства диагностики. Предназначены для автоматизации процессов диагностики программного и аппаратного обеспечения. Их используют для исправления ошибок и оптимизации работы компьютерной системы.

Программы инсталляции (установки). Предназначены для контроля за добавлением в текущую программную конфигурацию нового программного обеспечения. Они следят за состоянием и изменением программной среды, отслеживают и протоколируют образование новых связей. Простые средства управления установкой и удалением программ входят в состав операционной системы, но можно использовать и дополнительные служебные программы.

Средства коммуникации. Позволяют устанавливать соединение с удалёнными компьютерами, передают сообщения электронной почты, пересылают факсимильные сообщения и т. п.

Средства просмотра и воспроизведения. Предназначены для просмотра изображений и воспроизведения аудио- или видеофайлов.

Средства компьютерной безопасности. К ним относятся средства пассивной и активной защиты данных от повреждений, несанкционированного доступа, просмотра и изменения данных. Средства пассивной защиты — это служебные программы, предназначенные для резервного копирования. В качестве средств активной защиты применяется антивирусное

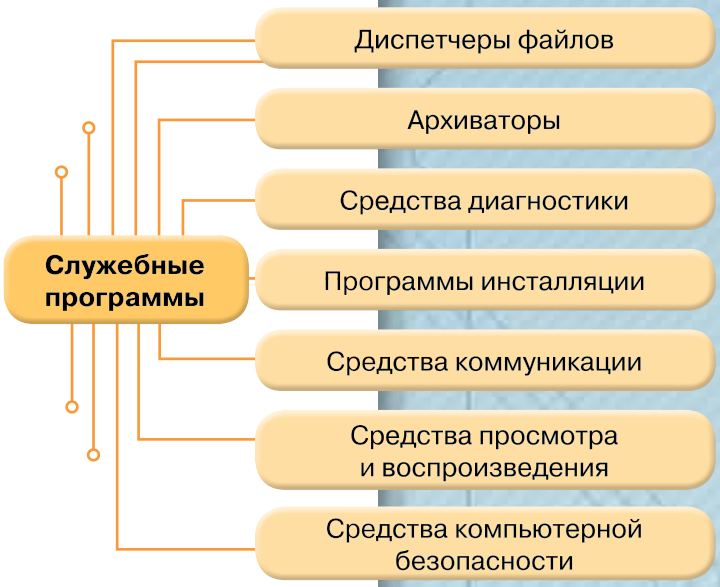


Рис. 5.7

программное обеспечение. Для защиты данных от несанкционированного доступа, их просмотра и изменения используются специальные системы, основанные на криптографии.

7. Когда программное обеспечение требует инсталляции и деинсталляции?

Большинство программ поставляется для продажи и распространения в упакованном виде. Для нормальной работы их нужно распаковать, а необходимые данные — правильно расположить в компьютере, учитывая отличия между компьютерами и настройками пользователя. В процессе установки выполняют определённые тесты на соответствие заданным требованиям, а компьютер настраивают для хранения файлов и данных, необходимых для правильной работы программы.

Процесс установки (инсталляция) программного обеспечения на компьютер пользователя может быть осуществлён с помощью:

- менеджера пакетов — специальной программы в составе операционной системы (например, *APT* в *Linux*, *Программы и средства* в *Microsoft Windows 7*);
- средства установки — специальной программы в составе самого программного обеспечения.

Установка программного обеспечения обычно предусматривает размещение всех необходимых файлов в соответствующих местах файловой системы, а также изменение и создание конфигурационных файлов. Менеджеры пакетов также выполняют контроль зависимостей, проверяя, есть ли в системе необходимые для работы данной программы средства, а в случае успешной установки регистрируют новое программное средство в перечне имеющихся.

Некоторые компьютерные программы созданы таким образом, что их устанавливают простым копированием файлов в нужное место. О таких программах говорят, что они не требуют инсталляции, они распространяются копированием.

Различают следующие виды инсталляции программ:

- обычная (англ. — *typical*, *normal*);
- минимальная (англ. — *minimum*);
- полная (англ. — *full*);
- выборочная (англ. — *custom*).

Удаление (деинсталляцию) программ необходимо выполнять с помощью системных или специальных программных средств. Простое удаление файлов не приводит к удалению программы из реестра установленного программного обеспечения.

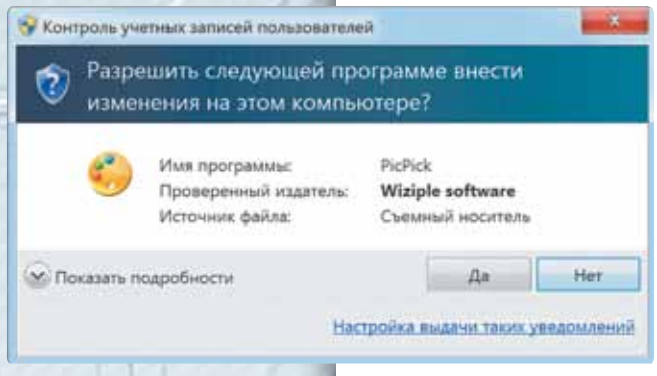


Рис. 5.8

ДЕЙСТВУЕМ

Упражнение 1. Инсталляция программы *PicPick*.
Задание. Инсталлируйте графический редактор *PicPick*.

1. В папке *Обеспечение компьютера* найдите инсталляционный пакет *picpick_inst.exe*.

2. В окне *Контроль учётных записей пользователя* (рис. 5.8) подтвердите запрос на установку программы на своём компьютере.

3. В окне установки программы прочитайте лицензионное соглашение. При необходимости воспользуйтесь онлайн-переводчиком. Нажмите кнопку *I Agree* (рис. 5.9) — подтверждение того, что вы принимаете условия соглашения и продолжаете инсталляцию программы.
4. Выберите место размещения программы: *C:\\Program Files(x86)\PicPick* и нажмите кнопку *Install.*
5. Дождитесь завершения процесса инсталляции.

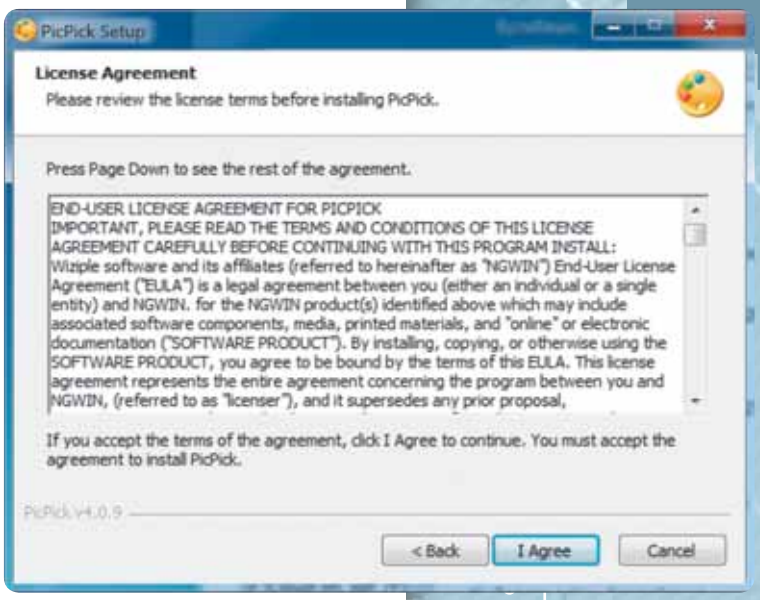


Рис. 5.9

8. В каких случаях говорят о проблемах совместимости программного обеспечения?

Сегодня существует большое количество различных производителей как аппаратной составляющей, так и программного обеспечения компьютера. Это привело к проблемам их совместимости — способности разных объектов взаимодействовать между собой. Если проблема аппаратной совместимости уже почти решена, то проблема совместимости программного обеспечения до сих пор ещё актуальна. Различают виды совместимости программного обеспечения на уровне:

- исполняемых файлов;
- программных кодов (программа может быть выполнена на разных компьютерах под управлением разных операционных систем);
- форматов файлов данных (программы могут отличаться интерфейсом, набором функций, но работать с одинаковыми документами);
- сетевой совместимости (способность программ обмениваться данными по сети).

9. Что такое форматирование носителей данных и с помощью какого программного обеспечения оно осуществляется?

К служебному программному обеспечению относятся также программы для работы с носителями данных. Наиболее используемой среди них является программа форматирования. Её обычно выполняют перед установкой всех программ на жёсткий диск, а также после приобретения дополнительного носителя данных.

Форматирование (англ. *formatting*) — процедура создания структур пустой файловой системы указанного типа — распределение дорожек жёсткого магнитного диска или другого носителя данных (например, флеш-накопителя или карты памяти) на физические или логические записи, которая выполняется перед первым использованием диска. Форматирование накопителя, содержащего данные, приводит к их потере.

При форматировании также могут проверяться целостность носителя и исправляться повреждения.

Устройства хранения данных, в частности флеш-накопители и карты памяти, могут поступать в продажу уже отформатированными.

ДЕЙСТВУЕМ

Упражнение 2. Форматирование флеш-накопителя.

Задание. Отформатируйте флеш-накопитель.

1. Присоедините флеш-накопитель к USB-разъёму. Проверьте, отображается ли название присоединённого устройства в списке устройств окна *Компьютер*.
2. Вызовите контекстное меню съёмного диска

. Выберите команду *Форматировать*.

3. В списке доступных файловых систем выберите *FAT32*. Задайте быстрый способ форматирования — очистку оглавления (рис. 5.10).
4. Нажмите кнопку *Начать*. Дождитесь, пока процесс форматирования не завершится. Он будет сопровождаться индикатором процесса выполнения задания в нижней части окна. Завершите форматирование, нажав кнопку *Закрыть*.
5. Проверьте, было ли удалено содержимое флеш-накопителя в процессе форматирования.

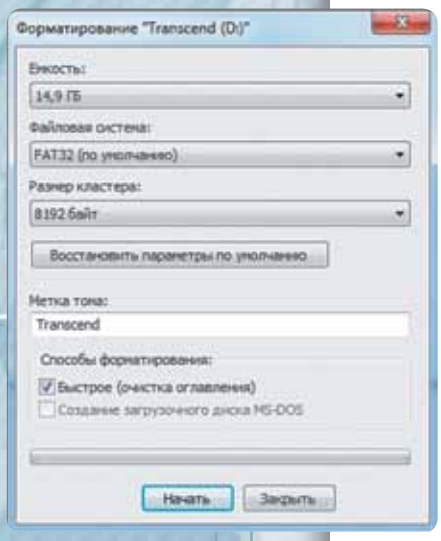


Рис. 5.10

10. Какие функции выполняют программы-архиваторы?

При передаче данных по компьютерной сети, а также при сохранении резервных копий файлов важен их объём. Поэтому часто применяют сжатие файлов. Сжимать можно не только один файл, но и папку, содержащую несколько файлов или папок. Результатом сжатия является запакованный файл, или архив. Создавать такие файлы и работать с ними позволяют специальные программы, которые называются архиваторами и программами резервного копирования.

Часто используются программы-архиваторы *7-Zip* (рис. 5.11), *WinRar*, *WinZip*. Архивы, созданные с помощью этих программ, имеют расширение соответственно *7z*, *rar*, *zip*.

К базовым функциям большинства современных архиваторов относятся:

- создание новых архивов;
- распаковка файлов из архивов (разархивирование);
- добавление файлов к архиву;
- создание самораспаковывающихся архивов;
- создание распределённых архивов на носителях небольшого объёма;
- тестирование целостности структуры архивов;
- полное или частичное восстановление повреждённых архивов;
- защита архивов от просмотра и несанкционированной модификации.

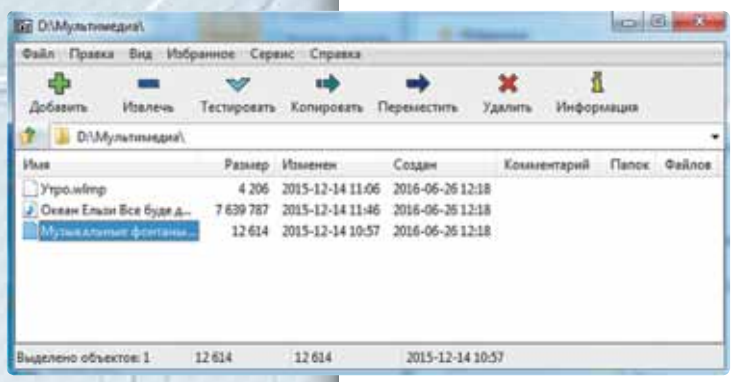


Рис. 5.11

Архивация предусматривает упаковку и сжатие данных. Упаковка и сжатие (компрессия) — не одно и то же. Упаковка — это слияние нескольких файлов или папок в единый файл, который называется **архивом**. Сжатие же — сокращение объёма исходного файла или группы файлов.

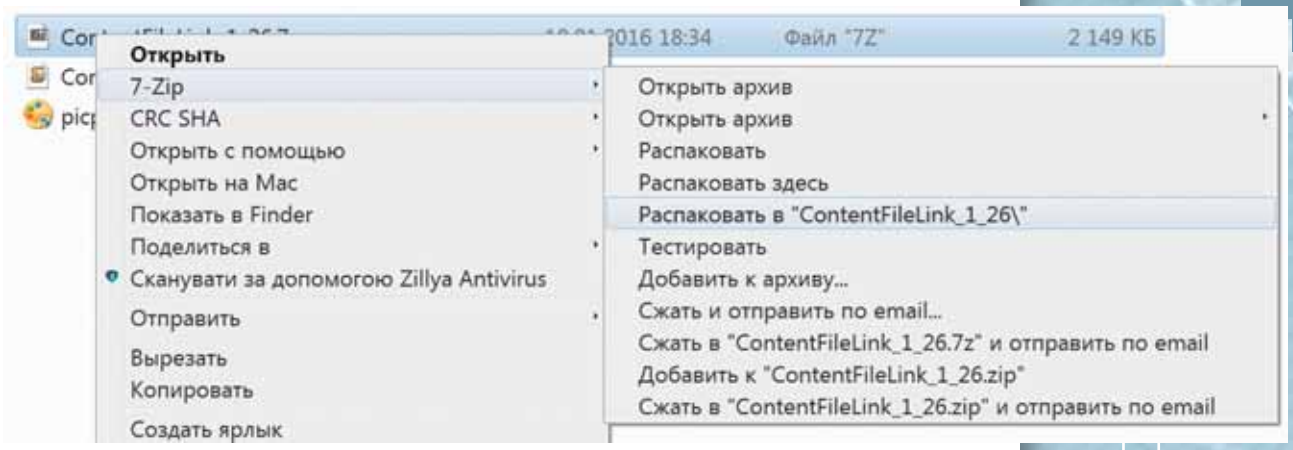


Рис. 5.12

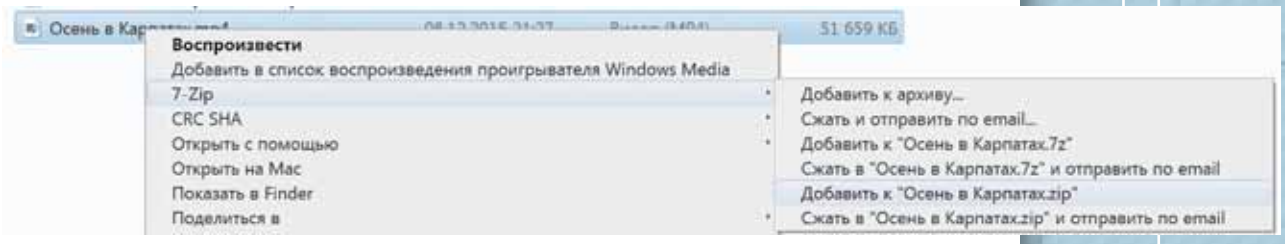


Рис. 5.13

В разных архиваторах применяют разные способы сжатия, поэтому объём файла архива по сравнению с исходным файлом может отличаться в зависимости от программы-архиватора, с помощью которой он был создан.

Быстро создать файл архива со свойствами, установленными по умолчанию, или распаковать архив можно с помощью команд контекстного меню. Если на компьютере установлена программа-архиватор, то команды для выполнения часто используемых операций с архивами выносятся в контекстное меню (рис. 5.12, 5.13).

Для добавления файлов к только что созданному или открытому архиву следует выбрать в программе-архиваторе команду *Добавить*, а затем найти и отметить нужные файлы и ещё раз воспользоваться командой *Добавить*, т. е. подтвердить выполнение ранее выбранной команды.

Для извлечения из архива одного или нескольких файлов сначала нужно найти архив. Потом с помощью программы-архиватора следует выделить те файлы, которые требуется распаковать, и выбрать команду распаковки, указав соответствующее место на диске для размещения файлов, которые будут раскрываться. При создании архива и занесении в него файлов и при его раскрытии остаются неизменными файлы-источники: при архивировании — файлы, которые сжимаются; при раскрытии архива — сжатые файлы.

ОС *Windows 7* содержит встроенные средства для работы с zip-архивами, которые ещё называют zip-папками. Отличить zip-папку от обычных папок можно по значку — он содержит «застёжку-молнию» (рис. 5.14).

Если дважды щёлкнуть на таком значке, то можно увидеть перечень сжатых файлов, а также в режиме *Таблицы* — объём исходных файлов и «упакованный» объём. Для просмотра файлов, содержащихся в zip-архиве, можно дважды

Интересно

Архивы формата ZIP некорректно работают с именами файлов и папок, содержащих символы кириллицы, — при распаковке таких архивов имена будут содержать другие символы. Поэтому при использовании zip-архивов нужно называть папки и файлы на латинице.

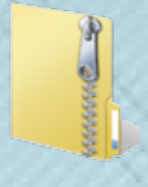


Рис. 5.14

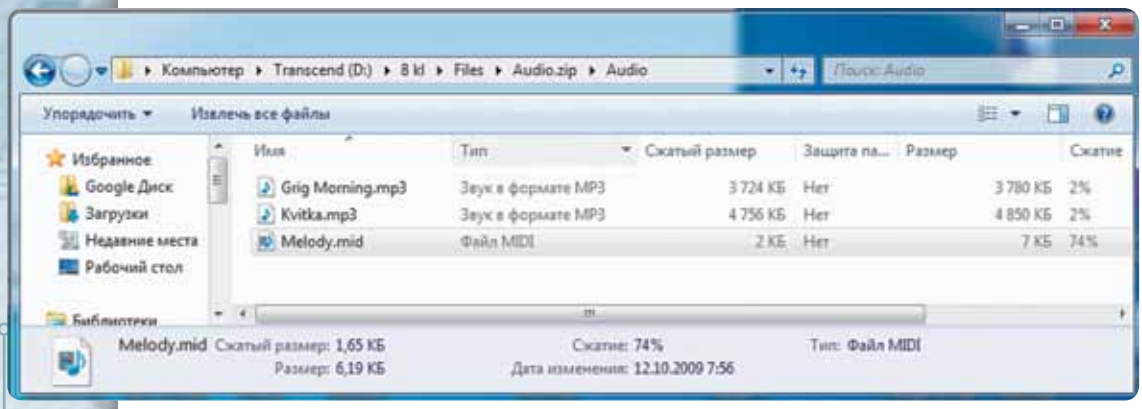


Рис. 5.15

щёлкнуть на значке соответствующего файла. Внесение изменений возможно только в распакованный файл. Чтобы разархивировать файлы из zip-архива, нужно его выделить и выбрать команду *Извлечь все файлы* (рис. 5.15).

В тех случаях, когда архивация выполняется для передачи пакета документов другому пользователю, следует предусмотреть наличие у него программного средства, необходимого для распаковки исходных данных из архива. Если у пользователя нет требуемой программы-архиватора, целесообразно создать самораспаковывающийся архив. Файл архива получает расширение *exe*, т. е. он является исполняемым файлом. Пользователь сможет запустить этот файл как обычную программу, после чего распаковка архива на его компьютере произойдёт автоматически.

Кроме того, у каждой из программ-архиваторов есть много дополнительных функций.

Интересно

Архив с расширением *exe* называют также SFX-архивом — от англ. *self-extracting archive* — самораспаковывающийся архив.

ДЕЙСТВУЕМ

Упражнение 3. Архивирование папки с помощью команд контекстного меню.

Задание. Заархивируйте папку *Методы сжатия*, находящуюся в папке *Обеспечение компьютера*, с помощью команд контекстного меню в формате *7z*.

1. Откройте папку *Обеспечение компьютера*, выделите в ней папку *Методы сжатия* и скопируйте её в папку *Компьютер и программы* своей структуры папок.
2. В своей структуре папок щёлкните правой кнопкой мыши на папке *Методы сжатия* и в контекстном меню выберите команду *7-zip/Добавить к "Методы сжатия.7z"*.
3. Убедитесь, что в папке *Компьютер и программы* появился соответствующий архив.
4. С помощью команды контекстного меню *Свойства* определите объём файла архива и объём исходной архивируемой папки.

Упражнение 4. Распаковка из архива отдельного файла с помощью архиватора *7-Zip*.

Задание. Распакуйте из архива *История развития вычислительной техники.7z*, находящегося в папке *Обеспечение компьютера*, файл *Ссылки на ресурсы в Интернете.txt* в свою папку.

1. Откройте папку *Обеспечение компьютера*.
2. Дважды щёлкните на файле архива *История развития вычислительной техники.7z*. Откроется окно программы-архиватора *7-zip* с содержимым выбранного файла архива.
3. В окне программы-архиватора *7-zip* дважды щёлкните на папке *История развития вычислительной техники*, хранящейся в запакованном виде.
4. Выделите файл *Ссылки на ресурсы в Интернете.txt* и нажмите кнопку *Извлечь* на панели инструментов окна *7-zip*.
5. В диалоговом окне *Извлечь*, пользуясь инструментом ... для обзора папок, укажите папку *Компьютер и программы* на своём компьютере, в которую необходимо распаковать файл, и нажмите кнопку *OK*.
6. Закройте все открытые окна.

Упражнение 5. Создание самораспаковывающегося архива с помощью архиватора 7-Zip.

Задание. Заархивируйте папку *Отдых в Украине* таким образом, чтобы получить самораспаковывающийся архив.

1. Загрузите программу-архиватор *7-Zip*.
2. В списке дисков и папок откройте папку *Обеспечение компьютера*.
3. Выделите папку *Отдых в Украине* и нажмите кнопку *Добавить* на панели инструментов окна *7-Zip*.
4. В диалоговом окне *Добавить* к архиву выберите инструмент ... и укажите папку *Компьютер и программы*, в которую будет сохранён архив. Имя архива по умолчанию будет совпадать с именем папки, которая добавляется к архиву.
5. В области *Опции* включите флажок *Создать SFX-архив* и нажмите кнопку *OK* (рис. 5.16).
6. Откройте окно папки, в которой был создан архив, и определите имя и расширение созданного файла.
7. Закройте все открытые окна.

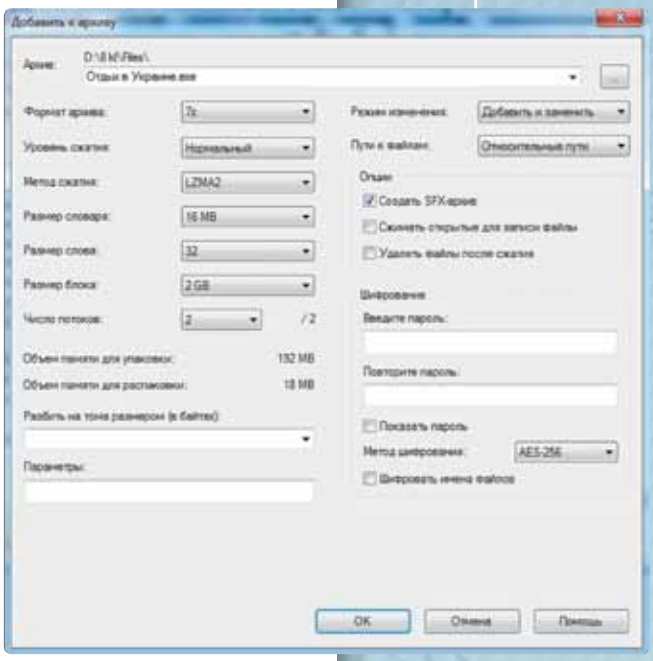


Рис. 5.16

11. В чём заключаются основные методы сжатия данных?

Характерной особенностью большинства форматов данных, с которыми традиционно работает пользователь, является определённая избыточность. Степень избыточности зависит от типа данных.

Пример избыточности — повторение в тексте фрагментов, например, некоторых слов или сочетаний букв в текстовых документах. Подобная избыточность обычно устраняется заменой повторяемых последовательностей более коротким значением — кодом. Допустим, в файле содержится много однотипных слов: *компьютер*, *компьютера*, *компьютерная*, *компьютеризация* и т. п. Если сочетание 9 букв «компьютер» заменить простой комбинацией символов «чц», то рассмотренный набор слов превратится в систему: «чц», «чца», «чцная», «чцизация» и т. п.

Другой вид избыточности связан с тем, что некоторые значения в сжимаемых данных встречаются чаще других. При этом можно заменять

часто встречаемые данные более короткими кодами, а те, что встречаются редко, — более длинными.

В видеофайлах избыточность, как правило, в несколько раз меньше, чем в графических, а в графических — меньше, чем в текстовых. Кроме того, степень избыточности данных зависит от принятой системы кодировки.

Существует большое количество алгоритмов сжатия данных, но все они работают по единому принципу — уменьшение избыточности данных в файле с помощью различных математических методов. В результате, в зависимости от совершенства алгоритма и типа исходного файла, его размер может существенно уменьшиться: типичным значением для документов является 40–50 % и более. Сверхнизкие показатели у видео- и аудиофайлов. И это вполне логично, ведь данные, хранящиеся в них, поддавались компрессии и практически не содержат избыточности.

Различают следующие виды сжатия:

- сжатие без потерь, при котором возможно извлечение исходных данных без искажений;
- сжатие с потерями — извлечение возможно с незначительными искажениями.

Сжатие без потерь используется, в частности, при обработке и хранении компьютерных программ и данных, когда такие потери недопустимы. Сжатие с потерями обычно применяется для уменьшения объёма звуковых, фото- и видеоданных.

ПОЛЕЗНЫЕ ССЫЛКИ

Сведения о различных алгоритмах сжатия и программах для архивации данных:
http://wiki.tntu.edu.ua/Архівація

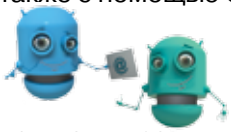
В основе работы программ-архиваторов лежит процедура поиска и перекодирования одинаковых фрагментов содержимого файла.

Каждая из программ-архиваторов работает по разным алгоритмам сжатия данных разных типов. В реальных программах-архиваторах процедура поиска и перекодирования данных происходит значительно сложнее.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 5* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ



1. Из перечня программ (рис. 5.17) составьте «пирамиду», отображающую порядок установки отмеченных программ на компьютере для поиска необходимых данных в Интернете и создания на их основе текстового сообщения на заданную тему. Обсудите в парах возможные варианты выполнения этого задания.

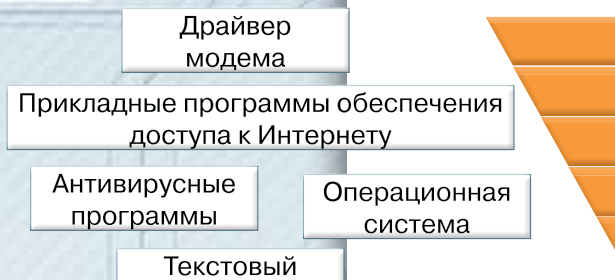


Рис. 5.17

2. Как раскрыть созданный архив? Сформулируйте обобщённое правило распаковки архива. Обсудите его в парах.

3. Почему на некоторых компьютерах нельзя играть в определённые компьютерные игры? Сформулируйте три возможные причины. Обсудите в парах.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Дополните классификационную схему программного обеспечения компьютера, содержащуюся в папке *Обеспечение компьютера* в файле *Программное обеспечение*, примерами программ с разными лицензиями. При необходимости воспользуйтесь сведениями из Интернета.
2. Найдите в Интернете данные о популярности современных операционных систем. На основе числовых данных в табличном процессоре постройте диаграммы для иллюстрации. Сделайте вывод о популярности операционной системы, которую вы используете для персонального компьютера и мобильного устройства.
3. В списке инсталлированных на вашем компьютере программ определите три программы, которые вы используете очень часто, и две, которые используете изредка. Установите, какой объём носителя данных освободится, если деинсталлировать одну из программ, которую вы вовсе не используете. Найдите в Интернете данные о назначении этой программы и сделайте вывод о целесообразности или нецелесообразности её деинсталляции.
4. При архивировании данных используют различные методы сжатия. Один из них — алгоритм Хаффмана. В основе алгоритма Хаффмана лежит идея кодировки битовыми группами. Рассмотрим простой пример, иллюстрирующий работу алгоритма Хаффмана. Допустим, задан текст, в котором буква *A* встречается 10 раз, буква *B* — 8 раз, *C* — 6 раз, *D* — 5 раз, *E* и *F* — по 4 раза. Один из возможных вариантов кодировки по алгоритму Хаффмана приведён в таблице 5.1.

Таблица 5.1

Символ	Частота вхождения	Битовый код	Символ	Частота вхождения	Битовый код
A	10	00	*D*	5	101
B	8	01	*E*	4	110
C	6	100	*F*	4	111

Составьте таблицы для кодирования по алгоритму Хаффмана слов «молоко», «кукареку». Найдите в Интернете дополнительные сведения об алгоритме Хаффмана и выясните, как определить длину кода заданного слова и коэффициент сжатия данных.

ИССЛЕДУЕМ

Определите, какие действия необходимо выполнить для деинсталляции программы. Для этого установите и деинсталлируйте пробную версию программы *Easy GIF Animator* по такому плану:

1. Загрузите пробную версию программы *Easy GIF Animator* с сайта *http://easy-gif-animator.ru.uptodown.com* и инсталлируйте её на компьютер.
2. Откройте *Главное меню*  и выберите вкладку *Панель управления*.
3. В списке категорий групп программ выберите *Программы/Программы и компоненты.*

4. Подождите, пока не будет сформирован список инсталлированных на компьютере программ. Найдите в списке программу *Easy GIF Animator v 6.2*, щёлкните на её названии в списке.
5. В списке команд, доступных для этой программы, выберите команду *Удалить*. Подтвердите запрос на удаление программы с компьютера, нажав кнопку *OK*.
6. Дождитесь завершения процесса деинсталляции. Убедитесь, что название деинсталлированной программы не отображается ни в списке программ, ни в главном меню компьютера.

6. ПРАКТИЧЕСКАЯ РАБОТА 3

АРХИВИРОВАНИЕ И РАЗАРХИВИРОВАНИЕ ДАННЫХ

ВСПОМНИТЕ

- Как создавать архивы разных типов;
- как добавлять данные к архивам, находить данные в архивах, извлекать данные из архивов;
- как обновлять архивы;
- как архивировать и разархивировать файлы и папки.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 3.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Открытие файла из архива (*5 баллов*)

Откройте файл архива с демонстрационной программой Международного конкурса по информатике и компьютерной грамотности «Бобёр» (*http://bober.net.ua/*) 2014 года. Сделайте вывод, можно ли это сделать из окна программы *7-Zip*, не распаковывая архив.

Задание 2. Извлечение файла из архива (*4 балла*)

В архиве *Песни Маруси Чурай*, содержащемся в папке *Обеспечение компьютера*, удалите файл с текстом другого автора.

Задание 3. Добавление файлов к архиву (*4 балла*)

Используя инструменты программы *7-Zip,* добавьте к архиву *Исторические песни*, содержащемуся в папке *Обеспечение компьютера,* файл *Песня о Байде.*

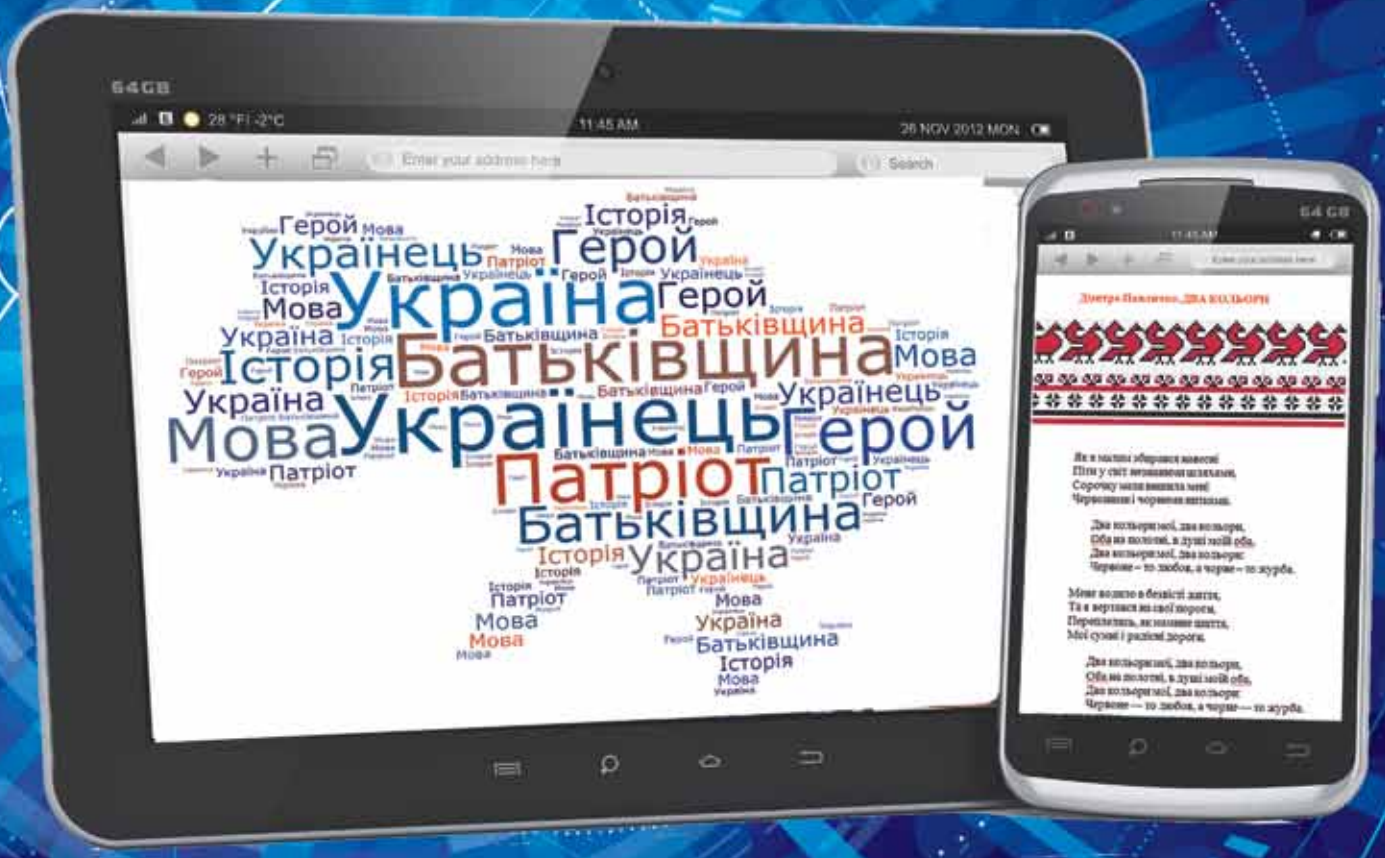
Задание 4. Создание архивов разных типов (*5 баллов*)

Для файла *Запорожское казачество*, содержащегося в папке *Обеспечение компьютера*, создайте rar-, zip- и SFX-архивы. Сравните размеры созданных архивов.

Задание 5. Обновление файла в архиве (*6 баллов*)

Дополните презентацию в архиве *Инструменты программы-архиватора*, содержащемся в папке *Обеспечение компьютера*, сведениями об использовании инструментов программы *7-Zip.*

Раздел 3



Работа с текстовыми данными

7. ТЕКСТОВЫЙ ДОКУМЕНТ И ЕГО ОБЪЕКТЫ

ВСПОМНИТЕ:

- как создавать, открывать и сохранять текстовый документ;
- как вводить текст с клавиатуры по правилам орфографии, пунктуации;
- как выделять и удалять, копировать и перемещать фрагменты текста;
- как форматировать символы и абзацы текста;
- как вставлять графические объекты в текстовый документ.

ВЫ УЗНАЕТЕ:

- какие существуют форматы файлов текстовых документов;
- как вставить в текст отсутствующие на клавиатуре символы;
- какие списки можно создать в текстовом документе;
- как разместить текст в несколько колонок;
- как вставить формулы в текстовый документ;
- как создавать и форматировать графические объекты в текстовом процессоре;
- как в текстовый документ вставлять таблицы;
- как редактировать и форматировать структуру таблицы;
- что такое непечатаемые символы.

ИЗУЧАЕМ

1. Какие существуют форматы файлов текстовых документов?

Вы уже знаете, что имена текстовых файлов, созданных в разных текстовых редакторах или процессорах, могут иметь разные расширения. Например, документ, созданный в текстовом редакторе *Блокнот*, имеет расширение *txt*, в текстовом процессоре *Microsoft Word* — *doc* или *docx*, в *LibreOffice Writer* — *odt*. Расширения файлов соответствуют форматам, определяющим способ организации данных в файле. Файлы разных форматов могут иметь разные значки. Это зависит от того, какие программы установлены на компьютере по умолчанию для работы с файлами соответствующих форматов (рис. 7.1).

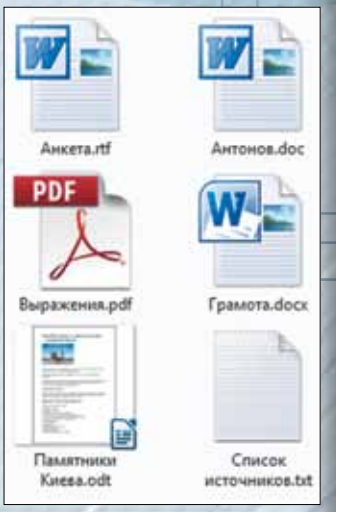


Рис. 7.1

Текстовый файл может содержать как форматированный, так и неформатированный текст.

Стандартным форматом для хранения неформатированного текста является *Обычный текст*, файлы этого формата имеют расширение *txt*. Все данные в таком файле представлены символами кодовой таблицы, которые без любых преобразований можно вводить с клавиатуры, выводить на экран или принтер.

Текстовые процессоры и издательские системы используют специально разработанные форматы файлов, содержащие не только текст, но и сведения о том, как он должен быть оформлен: какие параметры форматирования следует применить к определённым символам, фрагментам текста или абзацам и другим объектам, которые должны быть расположены вместе с текстом. Эти дополнительные сведения называются **разметкой текста**.

Универсальным форматом для хранения форматированного текста, поддерживаемым различными текстовыми процессорами, является формат RTF (от англ. *Rich Text Format* — расширенный текстовый формат).

Текстовые процессоры поддерживают работу с документами нескольких форматов. При сохранении текстового документа в среде текстового процессора кроме имени файла указывается также его тип, определяющий формат файла текстового документа. В раскрывающемся списке *Тип файла* указан

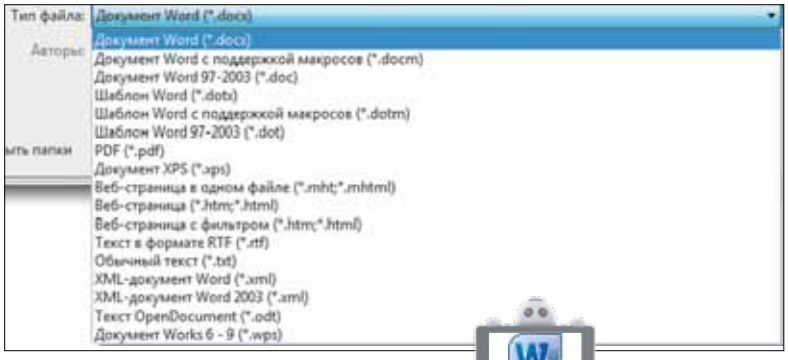


Рис. 7.2, *а*

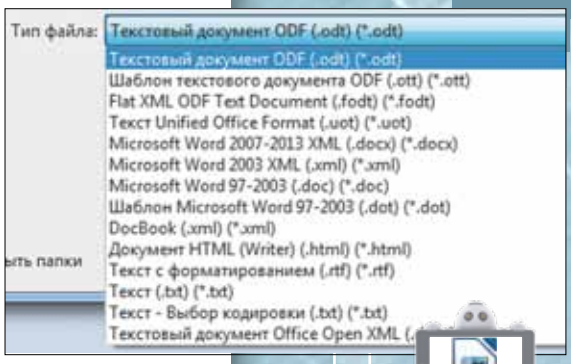


Рис. 7.2, *б*

перечень доступных форматов. Первым, как правило, указан собственный формат выбранного текстового процессора: *Документ Word* в *Microsoft Word* (рис. 7.2, *а*) и *Текстовый документ ODF* в *LibreOffice Writer* (рис. 7.2, *б*).

Для просмотра готовых текстовых документов используются форматы PDF и *Веб-страницы* — с расширениями *htm, html* (от англ. *HyperText Markup Language* — язык разметки гипертекстовых документов).

Текстовые документы в формате веб-страницы можно просматривать с помощью браузера. Их отличием от обычного текстового файла является то, что в HTML-документах используются специальные команды — тэги, определяющие правила форматирования документа. Однако текстовые документы, сохранённые в формате веб-страницы, не всегда будут отображаться именно в том виде, в каком были созданы — это зависит от размеров монитора, настроек браузера и других параметров. Для просмотра публикации в исходном виде используют формат PDF.

Интересно

Portable Document Format (PDF) — открытый формат файла, созданный и поддерживаемый компанией *Adobe Systems* для представления документов в независимом от устройства вывода и разрешающей способности виде. В декабре 2007 г. формат PDF был утверждён как международный стандарт.

2. Как вставить в текст отсутствующие на клавиатуре символы?

Вы уже умеете создавать текстовые документы, содержащие различные объекты: символы, которые вводятся с клавиатуры, абзацы, графические изображения, — а также редактировать и форматировать их.

Иногда в тексте необходимо использовать символы, которых нет на клавиатуре. Это могут быть знаки авторского права ©, математических операций сравнения ≠, ≤, ≥, ≈, греческие буквы α, β и другие символы.

Чтобы добавить в документ такие символы, в *Microsoft Word 2010* используется инструмент *Символ* на вкладке *Вставка* группы *Символы* (рис. 7.3). Некоторые часто используемые символы отображены в раскрывающемся списке. С помощью команды *Другие символы* открывается окно *Символ* (рис. 7.4), где можно выбирать разные символы. Список *Шрифт* содержит перечень доступных для выбора шрифтов. Например, шрифт *Wingdings* содержит символы в виде изображений книжки, телефона, письма, папки и т. п. (рис. 7.4).

Рис. 7.3

Некоторые символы можно вводить с помощью комбинаций клавиш или выбрать на вкладке *Специальные символы* окна *Символ* (рис. 7.5). В частности, символы *неразрывный дефис* (*Ctrl+Shift+–*) и *неразрывный пробел* (*Ctrl+Shift+Пробел*) используют тогда, когда слово или фразу, содержащую дефис или пробел, следует воспринимать как одно слово и не разрывать на две строки. Например, чтобы инициалы и фамилия *И. В. Карпенко* всегда были записаны в одной строке, нужно после инициалов использовать не обычный пробел, а неразрывный.

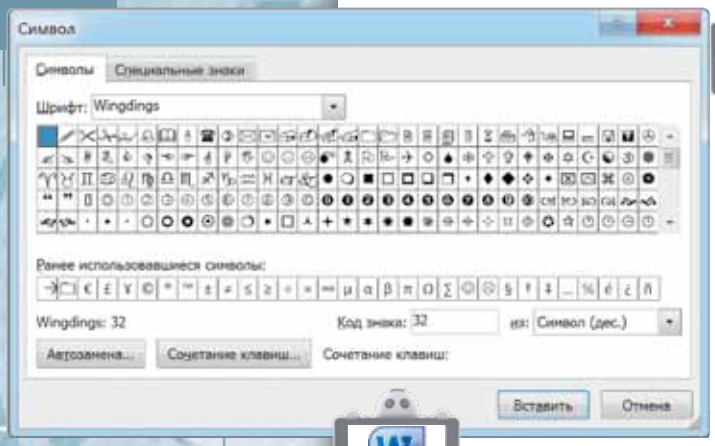


Рис. 7.4

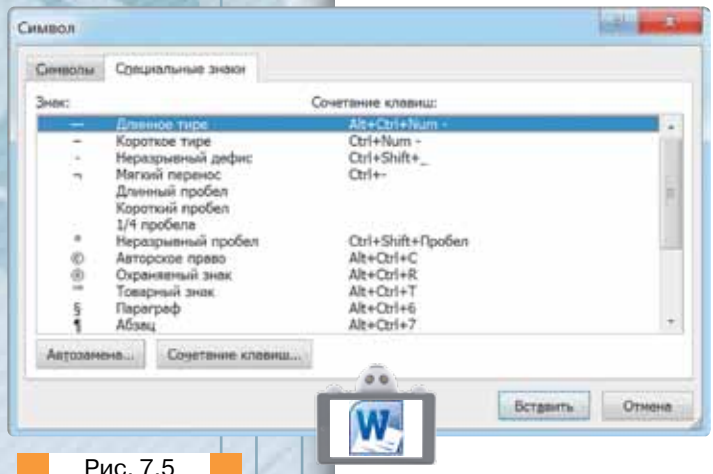


Рис. 7.5

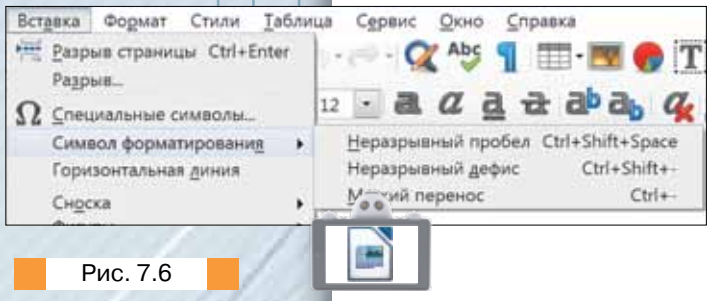


Рис. 7.6

В текстовом процессоре *LibreOffice Writer* вставлять символы можно с помощью инструмента *Специальные символы* Ω на панели инструментов *Стандартная* или команды меню *Вставка/Специальные символы*. Окно *Выбор символа* похоже на окно *Символ* в *Microsoft Word*. Неразрывный дефис или неразрывный пробел можно вставить с помощью комбинаций клавиш или команды *Вставка/Символ форматирования* (рис. 7.6).

Для вставленных символов, как и для символов, вводимых с клавиатуры, можно изменять значения параметров форматирования — цвет, размер, начертание и т. п.

3. Какие списки можно создать в текстовом документе?

Некоторые наборы данных документа можно оформить в виде списка, например, шаги инструкции выполнения задания, перечень обязанностей дежурного ученика, план сочинения. Текстовые процессоры содержат инструменты, с помощью которых можно создавать **нумерованные**, **маркированные** и **многоуровневые** списки. Нумерованный список создаётся для нумерации каждого элемента списка — это происходит автоматически, маркированный — если нумерация не важна, а следует только показать, что это отдельный элемент списка.

Если элементы списка уже внесены в документ и каждый из них находится в отдельном абзаце, то для создания списка нужно только выделить их и выбрать соответствующий инструмент. В текстовом процессоре *Microsoft Word 2010* инструменты для создания списков расположены на вкладке *Главная* в группе *Абзац*, в текстовом процессоре *LibreOffice Writer* — на панели инструментов *Форматирование*. В разных текстовых процессорах инструменты для создания списков похожи: *Маркеры* (*Маркированный список*), *Нумерация* (1 2 3) или *Многоуровневый список*. В списке, открываемом при выборе одного из инструментов, отображаются элементы соответствующей библиотеки (рис. 7.7–7.9). Чтобы выбрать нужный элемент, достаточно щёлкнуть на нём левой кнопкой мыши.

Чтобы изменить параметры форматирования списков в текстовом процессоре *Microsoft Word 2010*, нужно выбрать команду *Определить новый*

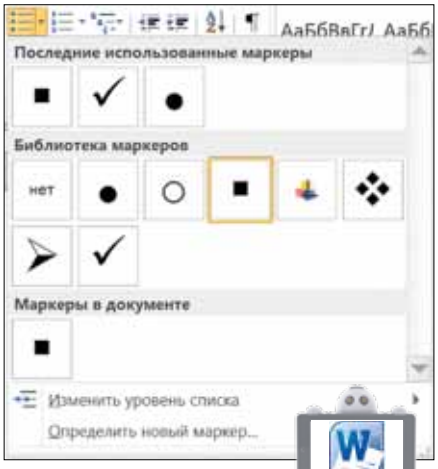


Рис. 7.7

Рис. 7.8

Рис. 7.9

маркер или *Определить новый числовой формат* в нижней части окна соответствующей библиотеки.

Изменить параметры форматирования списков или создать многоуровневый список в текстовом процессоре *LibreOffice Writer* можно с помощью инструментов, расположенных на вкладках окна *Маркеры и нумерация* (рис. 7.10). Открыть его можно с помощью команды *Ещё маркеры* или *Ещё нумерация* в нижней части окна библиотеки маркеров или нумерации. Выбрать форматы многоуровневого списка можно на вкладке *Структура* (рис. 7.10).

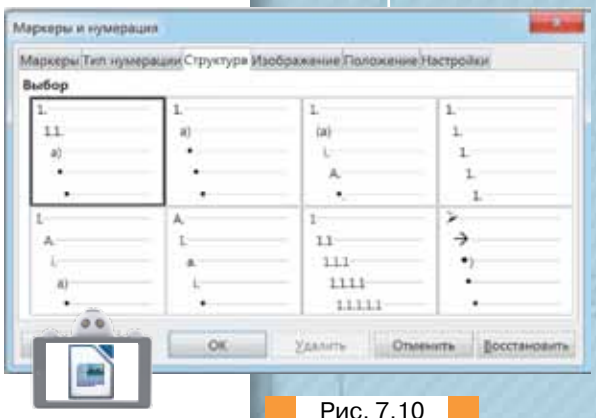


Рис. 7.10

Создать новый список, элементы которого ещё не содержатся в текстовом документе, можно аналогично: сначала необходимо установить курсор на новую строку, выбрать соответствующий инструмент для создания списка, ввести данные и нажать клавишу *Enter*. В новом абзаце создаётся новый элемент списка. Чтобы отказаться от нумерации или маркировки элемента списка, следует ещё раз выбрать инструмент, с помощью которого создавался список.

ДЕЙСТВУЕМ

Упражнение 1. Создание нумерованного и маркированного списков.
Задание. В документе *Подготовка реферата*, содержащемся в папке *Текстовый процессор*, создайте нумерованный список для списка печатных изданий и маркированный список для списка основных понятий реферата.

1. В своей структуре папок создайте папку *Тексты*.
2. В папке *Текстовый процессор* откройте документ *Подготовка реферата*.
3. Выделите элементы списка печатных изданий и выберите инструмент *Нумерация* и формат нумерации с точкой после номера (рис. 7.8).

4. Для списка печатных изданий этого документа внесите следующие изменения.
   - Удалите дублирующиеся элементы списка.
     Проанализируйте, как изменились номера других элементов такого нумерованного списка.
   - Отсортируйте список источников по алфавиту.
     Для перемещения элементов списка примените разные способы: перетягивание мышью, использование буфера обмена. Проанализируйте, как изменяются номера элементов списка.
5. Для списка основных понятий этого документа измените вид маркера на символ папки, для этого в библиотеке маркеров выберите команду *Определить новый маркер* в *Microsoft Word 2010* (команду *Ещё маркеры* в *LibreOffice Writer*), нажмите кнопку *Символ* (рис. 7.11), выберите шрифт *Wingdings,* символ папки и нажмите кнопку *OK*. Установите такие значения для форматирования элементов списка: отступ расположения маркера — 0,5 см; отступ расположения текста — 1,1 см.
6. Сохраните полученный документ в файле с тем же именем в папке *Тексты* своей структуры папок.

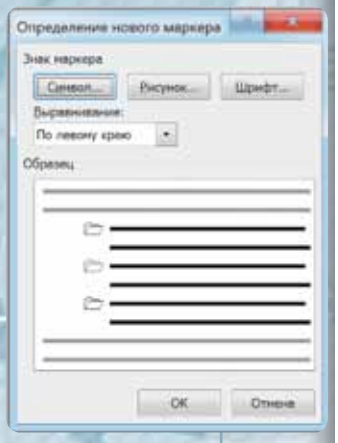


Рис. 7.11

4. Как разместить текст в несколько колонок?

В газетах и журналах можно увидеть текст, разбитый на несколько колонок. Такой текст можно подготовить в текстовом процессоре, при этом разные абзацы можно разбить на разное количество колонок.

Процесс подготовки документа, содержащего колонки, состоит из двух этапов: сначала вводят текст, а затем выделяют нужные абзацы и указывают количество колонок, в которые следует разместить выделенный фрагмент.

В *Microsoft Word 2010* для этого используют инструмент *Колонки* на вкладке *Разметка страницы* в группе *Параметры страницы*. Пользователю предлагается пять вариантов расположения выделенного текста в колонки (рис. 7.12).

Для форматирования колонок используют команду *Другие колонки*, при этом открывается окно *Колонки* (рис. 7.13). Все параметры настройки интуитивно понятны, к тому же в области *Образец* отображается схематический вид страницы.

В *LibreOffice Writer* для создания колонок используют команду *Формат/Колонки*. При этом появляется окно *Колонки* (рис. 7.14), в котором можно выбрать количество колонок и параметры их форматирования.

Интересно

Колонка — столбец печатного текста в газете, журнале и т. п.

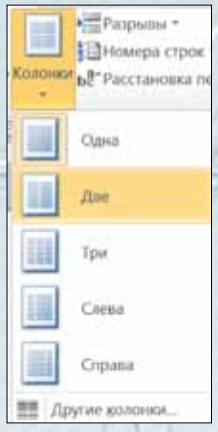


Рис. 7.12

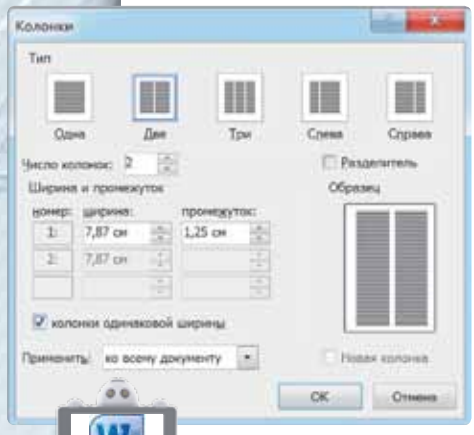


Рис. 7.13

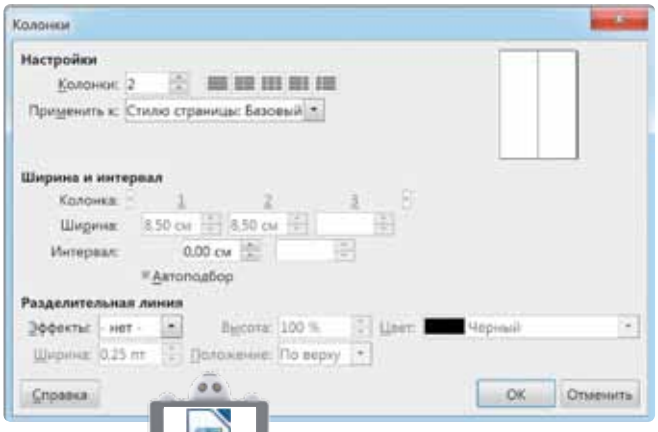


Рис. 7.14

ДЕЙСТВУЕМ

Упражнение 2. Стенгазета.

Задание. Представьте текст *Памяти героев Крут*, содержащийся в файле *История* папки *Текстовый процессор*, для размещения на информационном историческом стенде в школе на базе макета:

1. Загрузите текстовый документ *История*, находящийся в папке *Текстовый процессор*.
2. Выделите текст на первой странице. Воспользуйтесь инструментом *Колонки* на вкладке *Разметка страницы* в группе *Параметры страницы* текстового процессора *Microsoft Word 2010* или командой *Формат/Колонки* в *LibreOffice Writer*. Выберите способ создания колонок — *Справа*. Подтвердите выбор нажатием кнопки *OK*.
3. Выполните предыдущий шаг для второй и третьей страниц текста, каждый раз выбирая другой способ создания колонок, как показано в макете.
4. Сохраните файл с именем *Стенд* в папке *Тексты* в стандартном формате текстового процессора и в форматах PDF и RTF. Проверьте, влияет ли выбранный формат на способ размещения текста в колонки и как изменение формата влияет на размер файла. Сделайте выводы.

5. Как вставить формулы в текстовый документ?

Текстовые процессоры содержат встроенные инструменты для ввода с клавиатуры математических, физических или химических формул. Это упрощает процесс создания учебных и научных документов, содержащих формулы.

Рис. 7.15

В *Microsoft Word 2010* для этого используется инструмент *Формула* на вкладке *Вставка* в группе *Символы*. Раскрывающийся список содержит некоторые встроенные формулы, а также команду *Вставить новую формулу* (рис. 7.15), с помощью которой можно создавать собственные формулы.

После выбора этой команды становятся доступными инструменты *Конструктора* для работы с формулами, с помощью которых можно вставлять в поле формулы специальные символы (рис. 7.16) и структуры: дроби, индексы, корни, скобки и т. п. (рис. 7.17).

Раскрывающийся при выборе определённого инструмента список содержит набор разных шаблонов — образцов фрагмента формулы с использованием выбранной структуры. Например, с помощью инструмента *Радикал* можно выбрать шаблоны для записи квадратного или кубического корня, а также корня, степень которого необходимо ввести с клавиатуры (рис. 7.18).

Для конструирования формулы следует выбрать необходимые структуры и в выделенные поля ввести нужные символы. Символы можно вводить с клавиатуры или выбирать среди символов на вкладке *Конструктор*.

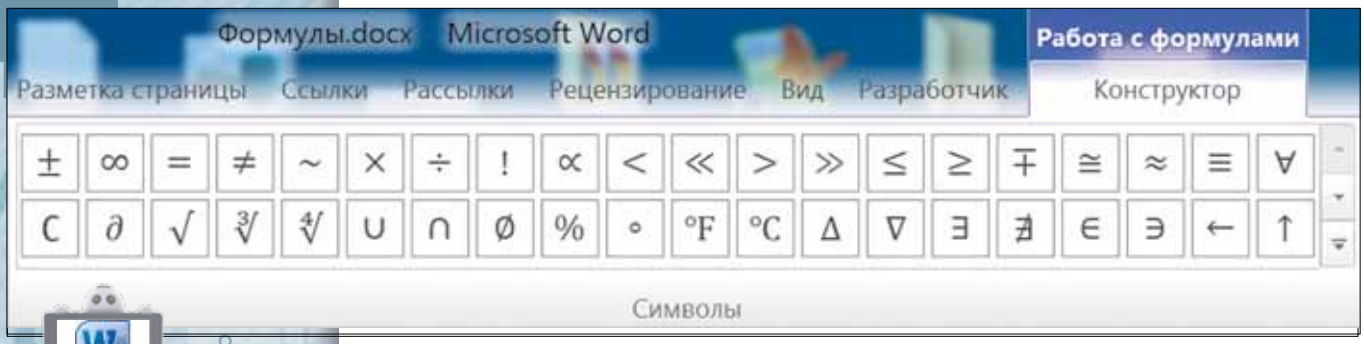


Рис. 7.16

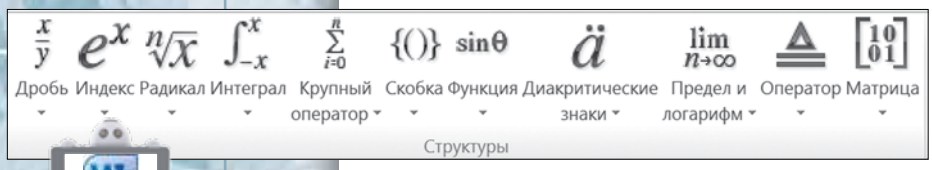


Рис. 7.17

Рис. 7.18

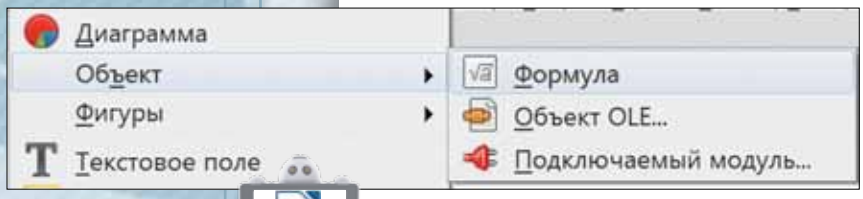


Рис. 7.19

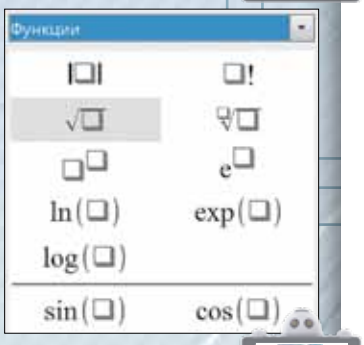


Рис. 7.20

Для завершения создания формулы нужно щёлкнуть левой кнопкой мыши за её пределами — т. е. завершить использование специального средства.

Чтобы отредактировать созданную формулу, необходимо дважды щёлкнуть мышью в её пределах, при этом станут доступными инструменты создания и редактирования формулы на вкладке *Конструктор*.

Для добавления формулы в текстовый документ в *LibreOffice Writer* используют команду *Вставка/Объект/Формула* (рис. 7.19).

Символы и структуры, которые можно использовать в формулах, собраны в список, открывающийся в левой части окна (рис. 7.20), и сгруппированы по категориям: отношения, функции, фигурные скобки и т. п.

ДЕЙСТВУЕМ

Упражнение 3. Создание формулы.

Задание. Создайте текстовый документ *Формула*, в который вставьте формулу

$$y = \frac{x^2 - 1}{|x| - 1}.$$

1. Загрузите текстовый процессор и создайте новый текстовый документ.
2. На вкладке *Вставка* выберите инструмент *Формула* в группе *Символы*.
3. Введите с клавиатуры начало формулы: $y =$.
4. На вкладке *Конструктор* в группе *Структуры* выберите инструмент *Дробь* и шаблон дроби с горизонтальной чертой $\frac{\square}{\square}$.
5. Выделите числитель дроби, на вкладке *Конструктор* в группе *Структуры* выберите инструмент *Индекс* и шаблон в виде степени.

6. Выделите основание степени, введите символ x, выделите показатель степени, введите символ *2*, установите указатель мыши в числителе дроби и введите *–1* (рис. 7.21).
7. Выделите знаменатель дроби. На вкладке *Конструктор* в группе *Структуры* выберите *Скобка* и шаблон в виде модуля $|\square|$. Выделите объект под модулем, введите символ x, установите указатель мыши в знаменателе после модуля, введите *–1*.
8. Щёлкните мышью за пределами формулы. Убедитесь, что созданная формула соответствует условию.
9. Сохраните результат в файле с именем *Формула* в папке *Тексты* своей структуры папок.

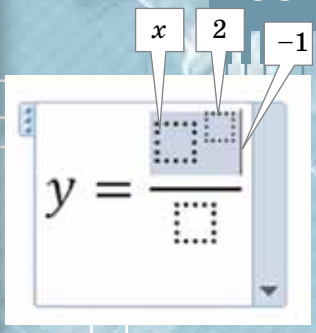


Рис. 7.21

6. Как создавать и форматировать графические объекты в текстовом процессоре?

Вы уже умеете добавлять в текстовый документ готовые графические изображения и форматировать их. Текстовые процессоры содержат встроенные средства, с помощью которых можно создавать изображения, состоящие из линий, прямоугольников и других фигур, непосредственно в текстовом процессоре.

В *Microsoft Word 2010* для создания изображений используют инструмент *Фигуры* из группы *Иллюстрации* на вкладке *Вставка*, в *LibreOffice Writer* — инструмент *Фигуры* в меню *Вставка*.

Microsoft Word 2010 содержит библиотеку готовых фигур, используемых для создания изображений, а также соответствующие инструменты в графическом редакторе (рис. 7.22, *а*). Для создания изображений в *LibreOffice Writer* можно воспользоваться инструментами из меню *Линия, Основные, Символы* (рис. 7.22, *б*).

Рис. 7.22, *а*

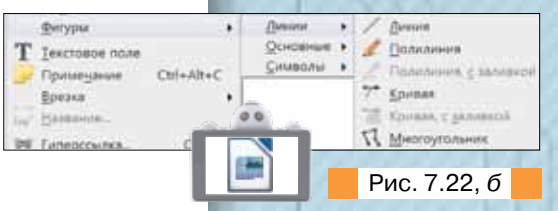


Рис. 7.22, *б*

Вставленные изображения можно форматировать: изменять свойства заливки фигуры и её контура, выбирать эффекты и стили фигур и т. п. Значения этих параметров в *Microsoft Word 2010* можно задавать с помощью инструментов на вкладке *Формат* в области *Стили фигур*, появляющейся при выделении созданной фигуры.

При размещении изображений в текстовом документе вместе с текстом, как и для готовых рисунков, важен способ их расположения относительно текста. Выбрать его можно с помощью инструментов *Положение* и *Обтекание текстом* из группы *Упорядочить* вкладки *Формат*. Инструменты из групп *Упорядочить* и *Размер* используют также для поворота, выравнивания, указания порядка размещения нескольких фигур и настройки размера фигуры. Изображения, состоящие из нескольких фигур, можно сгруппировать в один объект. Для этого сначала необходимо при нажатой клавише *Shift* последовательно выделить все фигуры, из которых состоит изображение. Затем необходимо выполнить команду *Группировать* с помощью

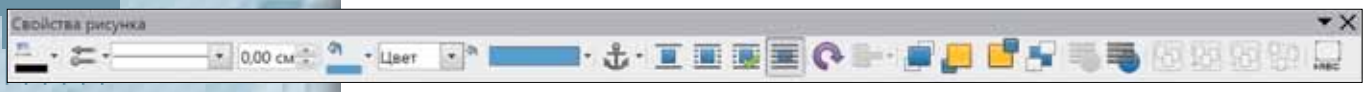


Рис. 7.23

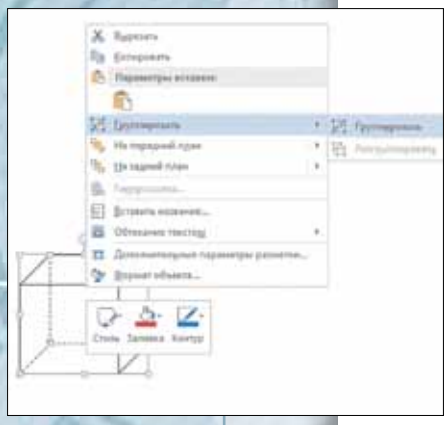


Рис. 7.24

соответствующего инструмента из группы *Упорядочить* вкладки *Формат* или контекстного меню (рис. 7.24). Когда несколько фигур сгруппированы в один объект, для составленного изображения легко изменять размеры и место размещения в текстовом документе, а также другие параметры.

В *LibreOffice Writer* изображение можно форматировать с помощью инструментов на панели *Свойства рисунка* (рис. 7.23), которая отображается вместо панели *Форматирование* при выделении созданной фигуры.

Контекстное меню выделенной фигуры или группы фигур в *LibreOffice Writer* содержит команды для изменения расположения объекта, выравнивания, обтекания текстом, группировки нескольких выделенных объектов и т. п.

ДЕЙСТВУЕМ

Упражнение 4. Создание изображения куба.

Задание. В новом документе *Куб* средствами текстового процессора создайте изображение куба. Добавьте текстовые надписи для обозначения вершин куба.

1. В папке *Тексты* своей структуры папок создайте новый документ с именем *Куб*.
2. На вкладке *Вставка* выберите инструмент *Фигуры* и из списка фигур в категории *Основные фигуры* выберите *Куб*. Выполните протягивание мышью в том месте документа, где вы планируете вставить изображение.
3. Выделите вставленное изображение и на вкладке *Формат* в группе *Стили фигур* выберите инструмент *Заливка фигуры*. Установите значение этого параметра *Нет заливки* (рис. 7.25), чтобы все грани куба не содержали заливки.
4. Среди фигур на вкладке *Вставка* выберите *Линия* и добавьте к изображению куба три линии, отображающие его невидимые рёбра.
5. Последовательно выделите каждую из линий и с помощью инструмента *Контур фигуры* на вкладке *Формат* измените значение параметра *Штрихи* на пунктирную линию.
6. Нажмите на клавиатуре клавишу *Shift*, удерживая её, выделите линии и фигуру, из которых состоит куб. Щёлкните правой кнопкой мыши на одной из выделенных границ и в контекстном меню выберите команду *Группировать/Группировать* (рис. 7.24).
7. С помощью инструмента *Надпись* на вкладке *Вставка* добавьте рядом с вершинами куба надписи с именами вершин: *A*, *B*, *C*, *D*, *A*', *B*', *C*', *D*'.
8. Сохраните результаты работы.

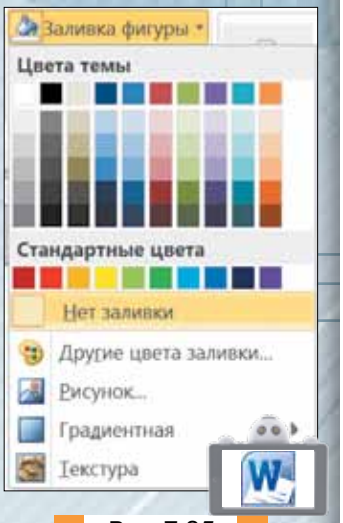


Рис. 7.25

7. Как в текстовый документ вставлять таблицы?

Для представления данных в структурированном виде в текстовых документах используются таблицы. Те отношения между данными, которые могут быть не замечены при записи «в одну строку», становятся наглядными в виде

таблицы. Например, в таком виде удобно представлять данные о проведении опытов при разных условиях на уроках физики.

Как и при работе с электронными таблицами, в текстовом процессоре при работе с таблицами различают следующие объекты: таблицы, строки, столбцы, ячейки. Каждый из объектов имеет свойства и список действий, которые можно с ним выполнять.

Для вставки таблицы в текстовый документ в *Microsoft Word 2010* используют инструмент *Таблица* 

на вкладке *Вставка*.

С его помощью можно создавать таблицу разными способами.

1. Выделить мышью количество строк и столбцов таблицы (рис. 7.26).

2. Выбрать команду *Вставить таблицу* и указать нужное количество столбцов и строк в окне *Вставка таблицы* (рис. 7.27).

3. С помощью команды *Нарисовать таблицу* можно легко создать таблицу сложной структуры, например, с ячейками разной высоты или разным количеством столбцов в строке. При этом используется метод, похожий на рисование таблицы от руки. Указатель мыши превращается в карандаш. С помощью этого карандаша можно нарисовать границы ячеек, строк и столбцов — так же, как на листе бумаги.

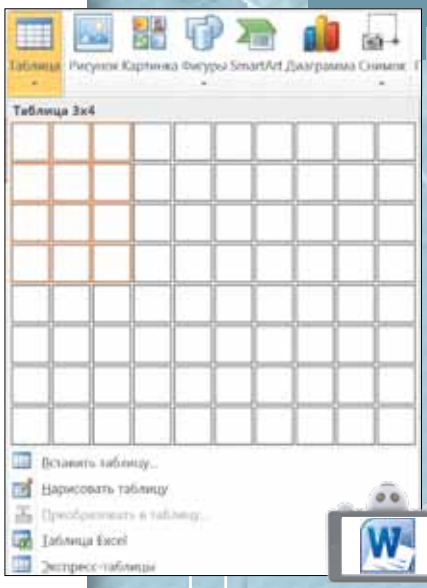


Рис. 7.26

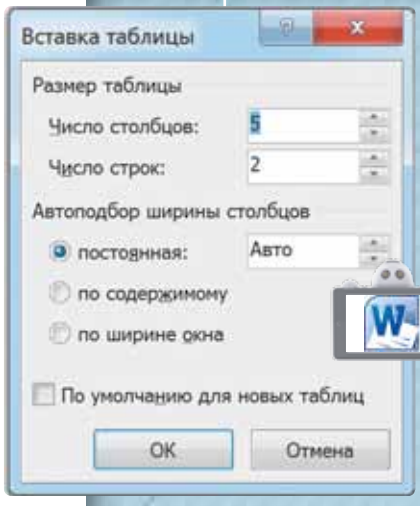


Рис. 7.27

Аналогично в текстовом процессоре *LibreOffice Writer* таблицу можно вставить с помощью инструмента *Вставить таблицу*  (рис. 7.28) или команды *Таблица/Вставить таблицу*, которая вызывает окно *Вставить таблицу*.

8. Как редактировать и форматировать структуру таблицы?

Созданную таблицу можно форматировать, а также вносить изменения в её структуру: объединять ячейки, вставлять и удалять столбцы или строки, а также изменять их размер.

Если необходимо применить одинаковые параметры форматирования ко всем ячейкам таблицы, сначала её нужно выделить. Чтобы быстро выделить всю таблицу, следует навести указатель мыши на любую ячейку таблицы и щёлкнуть на значке  , который появляется рядом с левым верхним углом таблицы.

Когда текстовый курсор находится в одной из ячеек таблицы, становятся доступными инструменты для редактирования и форматирования таблицы и её объектов.

В *Microsoft Word 2010* для этого можно использовать разные инструменты в области *Работа с таблицами* на вкладках *Конструктор* и *Макет* или команды контекстного меню.

С помощью инструментов вкладки *Конструктор* (рис. 7.29) можно выбрать стиль таблицы, изменить значения параметров её форматирования — цвет и стиль заливки, установить обрамление отдельных выделенных ячеек, изменить толщину и цвет границ и т. п. Настройка этих параметров похожа на аналогичные действия в электронных таблицах.

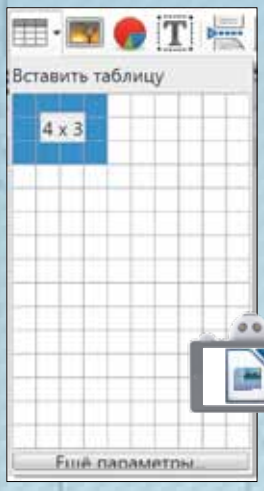


Рис. 7.28

Рис. 7.29

Редактировать структуру таблицы — вставлять и удалять строки или столбцы, объединять или разделять ячейки — можно с помощью инструментов вкладки *Макет* (рис. 7.30) и команд контекстного меню. Для объединения ячеек таблицы их нужно выделить, затем выбрать соответствующий инструмент на вкладке *Макет* или команду *Объединить ячейки* в контекстном меню. С помощью инструмента *Направление текста* можно размещать текст в ячейках горизонтально или вертикально.

Рис. 7.30

В *LibreOffice Writer* редактирование и форматирование структуры таблицы и её объектов можно выполнить с помощью инструментов на панели *Таблицы* (рис. 7.31), команд меню *Таблица* или контекстного меню.

Рис. 7.31

Сначала все столбцы имеют одинаковую ширину, а строки — одинаковую высоту. Можно вручную изменить ширину отдельного столбца или позволить системе автоматически выбрать ширину каждого столбца в зависимости от его содержимого.

Изменить ширину столбцов и высоту строк можно, перемещая маркеры границ таблицы на горизонтальной или вертикальной линейке масштабирования (рис. 7.32).

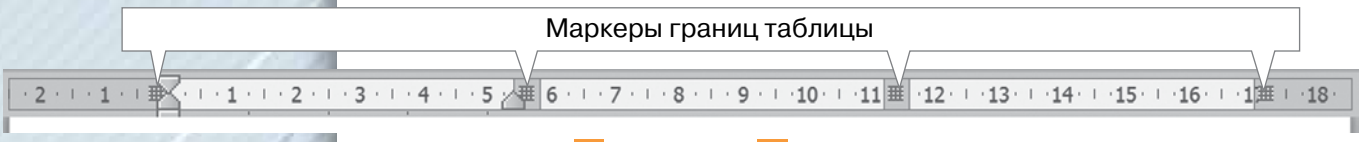


Рис. 7.32

Для изменения размеров столбцов или строк можно также навести указатель мыши на соответствующую границу в таблице и, когда он примет вид линии с двунаправленной стрелкой, выполнить протягивание мышью в нужном направлении.

ДЕЙСТВУЕМ

Упражнение 5. Создание таблицы и настройка её параметров.

Задание. Используя таблицу и изображение, создайте объявление о поиске хозяев для щенка (рис. 7.33).

1. В папке *Тексты* своей структуры папок создайте новый документ, который будет содержать объявление о поиске хозяев для щенка.

2. В созданном документе вставьте таблицу 2 × 5 одним из известных вам способов.
3. Объедините все ячейки первой строки таблицы.
4. Введите в ячейку первой строки текст «*Отдам в хорошие руки маленького щенка*» и выберите следующие параметры форматирования: шрифт — *Times New Roman*, размер — *12*, стиль начертания — *полужирный*, *курсив*; выравнивание — *по центру*.
5. Вставьте в эту же ячейку фото щенка, находящееся в файле *Щенок.jpg* в папке *Текстовый процессор*. Измените его размер, установите обтекание текстом *Вокруг рамки*, чтобы разместить его по образцу (рис. 7.33).
6. В первом столбце второй строки введите текст «тел. 050-0-55-55 Спросить Марию», выделите его и на вкладке *Макет* в группе *Выравнивание* выберите инструмент *Направление текста*. Выровняйте текст по центру и выберите размер символов — *10*.
7. Скопируйте текст в другие столбцы второй строки таблицы.
8. Сохраните документ с именем *Объявление* в собственном формате документов текстового процессора и в формате PDF.

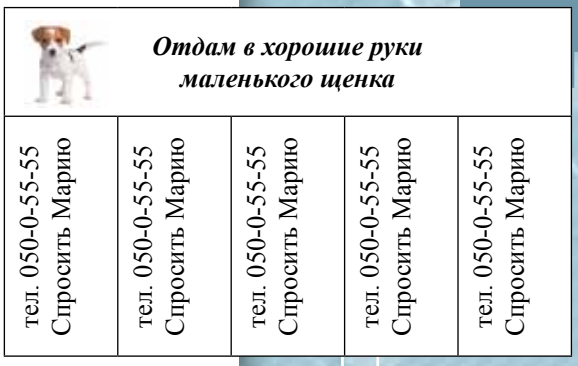


Рис. 7.33

9. Что такое непечатаемые символы?

Некоторые символы определяют текстовые объекты документа, но при печати никак не отображаются. Это такие символы, как пробел, конец абзаца, конец ячейки таблицы и т. п.: с помощью пробела одно слово отделяется от другого, при нажатии клавиши *Enter* создаётся новый абзац, на пересечении строк и столбцов таблицы образуется ячейка.

Иногда необходимо осуществить переход на новую строку раньше, без создания нового абзаца. В этом случае применяют принудительный разрыв строки, выполняемый нажатием комбинации клавиш *Shift* + *Enter*. Такой приём часто применяется при вводе текстов стихотворений или песен, когда желательно, чтобы каждые 4 строки (или другое количество строк, в зависимости от необходимых для этого текста правил оформления) составляли один абзац. Для обозначения перехода на новую строку также используется непечатаемый символ.

При подготовке документа можно включить режим, в котором непечатаемые символы будут отображены. Это позволит не только увидеть объекты документа, но и отследить некоторые возможные ошибки, чтобы потом их исправить. Ведь в тексте, например, между словами могло быть введено несколько пробелов вместо одного, или точка в конце предложения была оторвана от последней написанной буквы и т. п.

Для включения режима отображения непечатаемых символов используется инструмент *Отобразить все знаки* (*Непечатаемые символы* ), расположенный на вкладке *Главная* в группе *Абзац* в *Microsoft Word 2010* (на *Стандартной* панели инструментов в *Libre Office Writer*).

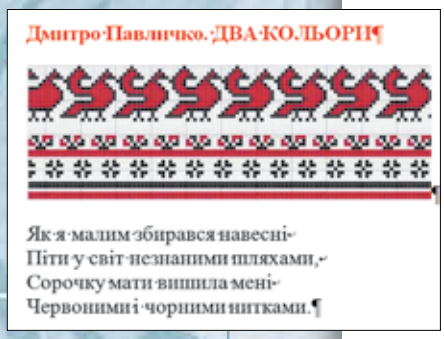
В режиме отображения непечатаемых символов в документе можно увидеть различные символы (табл. 7.1).

Таблица 7.1

Непечатаемый символ	Описание
·	Пробел
°	Неразрывный пробел
¶	Конец абзаца
↵	Переход на новую строку, без создания нового абзаца
¤	Конец ячейки таблицы

ДЕЙСТВУЕМ

Упражнение 6. Изменение вида документа в режиме отображения непечатаемых символов.


Дмитро·Павличко.·ДВА·КОЛЬОРИ¶

Як·я·малим·збирався·навесні·
Піти·у·світ·незнаними·шляхами,·
Сорочку·мати·вишила·мені·
Червоними·і·чорними·нитками.¶

Рис. 7.34

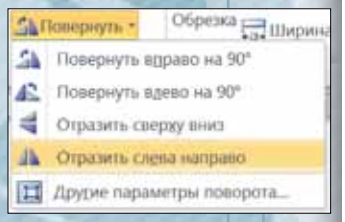


Рис. 7.35

Задание. Просмотрите документ *Два кольори*, находящийся в папке *Текстовый процессор*, в режиме отображения непечатаемых символов и определите, какие клавиши или комбинации клавиш использовались для завершения каждой строки. Добавьте после названия стихотворения изображение, содержащееся в файле *Вышивка.jpg*, обрежьте его и отразите слева направо, чтобы документ выглядел, как на рисунке 7.34.

1. В папке *Текстовый процессор* откройте документ *Два кольори*.
2. Включите режим отображения непечатаемых символов с помощью инструмента *Отобразить все знаки* ¶.
3. Определите, нажатием какой клавиши или комбинации клавиш завершался ввод каждой из строк стихотворения Д. Павлычко «Два кольори». Сколько абзацев в документе?
4. Добавьте после названия стихотворения новый абзац, вставьте в него изображение, содержащееся в файле *Вышивка.jpg* в папке *Текстовый процессор*.
5. Выделите изображение, на вкладке *Формат* выберите инструмент *Обрезка*. Наведите указатель мыши на маркер обрезки в нижней части изображения и выполните протягивание, чтобы получить фрагмент изображения, как на рисунке 7.34.
6. На вкладке *Формат* выберите инструмент *Повернуть* и команду *Отразить слева направо* (рис. 7.35).
7. Сохраните файл *Два кольори* в папке *Тексты* своей структуры папок.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 7* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Сравните значки файлов, сохранённых в разных форматах (рис. 7.1). Обсудите в парах гипотезы о причинах такого изображения.
2. Сравните параметры изменений нумерованного и маркированного списков (рис. 7.8, 7.9), найдите общее и отличия. Обсудите в парах.
3. Приведите примеры использования колонок в текстовых документах. Обсудите, какое количество колонок лучше использовать и при каких условиях.
4. Чем отличаются специальные и непечатаемые символы? Когда они используются? Обсудите в парах.
5. Поиграйте в игру «Формула» — по очереди приводите примеры математических, химических и физических формул, которые невозможно ввести с клавиатуры, а необходимо применять средства создания формул.

ПОЛЕЗНЫЕ ССЫЛКИ

Работа с объектами в *Microsoft Word 2010*: *http://e-helper.com.ua/msword_lections*

6. Можно ли переместить в другое место вставленную в документ таблицу? Как это сделать? Можно ли переместить часть таблицы? Обсудите в парах.
7. Можно ли содержимое отдельной ячейки таблицы отформатировать иначе, чем другие части таблицы? Обсудите в парах.
8. Можно ли определить лучший способ создания таблицы средствами текстового процессора? Приведите примеры ситуаций, когда целесообразно создавать таблицу определённым способом. Обсудите в парах.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. В папке *Текстовый процессор\Формат* размещены текстовые файлы, сохранённые в разных форматах. Они содержат одинаковые сведения. Сравните формат и объём каждого файла. Постройте для сравнения гистограмму в табличном процессоре, сделайте выводы. Загрузите в текстовом процессоре каждый из файлов, просмотрите и сделайте выводы.
2. Откройте файл *О форматах*, находящийся в папке *Текстовый процессор*, и разместите текст в две колонки. Сохраните этот файл в папке *Тексты* своей структуры папок в форматах DOC, DOCX, ODT, RTF и PDF. Определите объём файла в разных форматах. В каком из пяти форматов объём файла наименьший?
3. Создайте в текстовом процессоре блок-схему алгоритма определения спряжения глагола. Воспользуйтесь средствами для создания фигур. В алгоритме используйте команды, представленные на рисунке 7.36. Начните с команды, изображённой на рисунке 7.36, *а*.

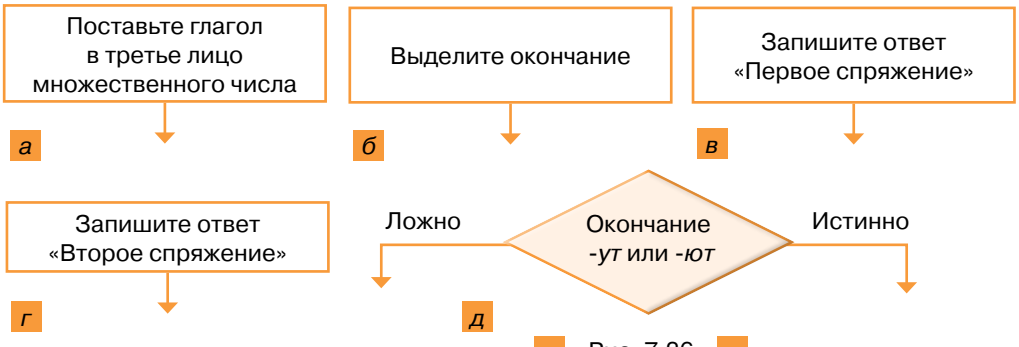


Рис. 7.36

4. Создайте документ *Календарь*, в котором с помощью инструментов рисования и установки параметров таблиц в текстовом документе оформите в виде таблицы календарь на текущий месяц и внесите в соответствующие ячейки дни рождения друзей и наиболее важные события на месяц. Например, шаблон календаря на сентябрь 2016 г. показан на рисунке 7.37. Все ячейки сделайте одинакового размера; примените к таблице один из стилей оформления, добавьте в ячейки таблицы для обозначения событий изображения из коллекции клипов.
5. Представьте сведения, содержащиеся в файле *Изобретения украинцев* в папке *Текстовый процессор*, в виде таблицы из пяти строк и пяти столбцов следующей структуры:

№	Изобретение	Год создания	Автор, портрет	Изображение изобретения

Сентябрь 2016 г.	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота	Воскресенье
			1	2	3	4	5
6	7	8	9	10	11	12	
13	14	15	16	17	18	19	
20	21	22	23	24	25	26	
27	28	29	30				

Рис. 7.37

Для ввода данных в таблицу воспользуйтесь буфером обмена при перемещении необходимых данных. Найдите в Интернете и добавьте изображения указанных изобретений. При необходимости измените размеры изображений или обрежьте их.

6. Создайте текстовый документ, содержащий такие формулы:

1) $(x_2 - x_1)^2 \geq 0$;

2) $y = \dfrac{1}{x - \dfrac{1}{x}}$;

3) $\begin{cases} 2x + 5y = 13, \\ 3x - 5y = -3; \end{cases}$

4) $Cu(OH)_2 \xrightarrow{t} CuO + H_2O$;

5) $\rho = \dfrac{m}{V}$.

ИССЛЕДУЕМ

Исследуйте, как в текстовом документе таблица может размещаться относительно текста. Для этого из контекстного меню таблицы выберите команду *Свойства таблицы* и в открывшемся окне определите возможные значения параметров *Выравнивание* и *Обтекание* на вкладке *Таблица*.

8. ПРАКТИЧЕСКАЯ РАБОТА 4

СОЗДАНИЕ ТЕКСТОВОГО ДОКУМЕНТА, СОДЕРЖАЩЕГО ОБЪЕКТЫ РАЗНЫХ ТИПОВ

ВСПОМНИТЕ

- Как создавать текстовый документ;
- как в текстовом документе создавать списки и колонки;
- как добавлять в текстовый документ различные объекты: символы, изображения, таблицы, формулы;
- как форматировать объекты текстового документа.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 4.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Процессор (*8 баллов*)

В документе *Процессор*, находящемся в папке *Текстовый процессор*, первых три абзаца текста разместите в две колонки и добавьте изображения по образцу. Для списка характеристик процессора измените маркер на ▪ (рис. 8.1):

Процессор называют электронным «мозгом» компьютера. Он предназначен для автоматической обработки и преобразования данных по заранее введенным программам и для управления работой всех устройств компьютера. От его вычислительной мощности в основном и зависит производительность компьютера.

Процессор — это микросхема, которая создается на полупроводниковом кристалле (или нескольких кристаллах) путем применения сложной микроэлектронной технологии. Различные операции в процессоре выполняются по специальным командам. Команды для процессора записывают в компьютерной программе.

Во время работы процессор довольно сильно нагревается, поэтому на него устанавливают систему охлаждения — вентилятор, который называют *кулером*.

Основными характеристиками процессоров являются:

- *тип* — в соответствии с фирмой-производителем различают процессоры Intel (Pentium, Celeron, Core2 Duo и т.п.), AMD (AMD64, Duron, Athlon и т.д.) и другие;
- *тактовая частота* — определяет количество выполняемых элементарных операций за одну секунду, то есть быстродействие процессора; тактовая частота современных процессоров измеряется в гигагерцах (ГГц), уже разработаны процессоры с частотой более 3 ГГц;
- *разрядность* — максимальная длина двоичного кода, который может обрабатываться или передаваться процессором; чем выше разрядность, тем выше мощность процессора;
- *кэш-память* — это внутренняя память процессора, которая позволяет сохранять промежуточные данные.

Рис. 8.1

Задание 2. Рациональные выражения (*8 баллов*)

Создайте новый документ *Выражения*, в котором с помощью средств создания формул и нумерованных списков введите текст задания:

Задание.

Какому из приведённых выражений тождественно равна дробь $\frac{6x^2}{24x}$?

1) $\frac{x^2}{4}$; 2) $\frac{12x^3}{48x}$; 3) $\frac{x}{4}$; 4) $\frac{3x^4}{24x}$.

Задание 3. Лабиринт (7 баллов)

Создайте новый документ *Лабиринт*, в котором с помощью средств создания таблиц и их обрамления постройте лабиринт по образцу. Добавьте и отформатируйте изображение, содержащееся в файле *Мышка.png* в папке *Текстовый процессор* (рис. 8.2).

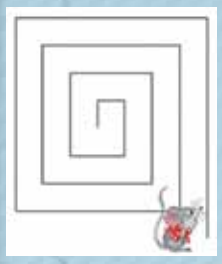


Рис. 8.2

Задание 4. Семь чудес Украины (*8 баллов*)

Дополните и отформатируйте документ *Семь чудес Украины*, находящийся в папке *Текстовый процессор*.

Задание 5. Приглашение (*10 баллов*)

Создайте новый документ *Приглашение*, в котором подготовьте приглашение на выставку «Я тобой, Украина, живу...». Примените таблицу без оформления для форматирования данных по образцу (рис. 8.3).

Рисунок или эмблема компании	Адрес
Текст приглашения	
Телефон	
Электронная почта	

Рис. 8.3

9. РАБОТА СО СЛОЖНЫМИ ТЕКСТОВЫМИ ДОКУМЕНТАМИ

ВСПОМНИТЕ:

- какие параметры форматирования символов и абзацев используются при оформлении текста;
- как добавить нумерацию страниц;
- как напечатать текстовый документ;
- как выбрать режим работы с документом;
- какие действия выполняют с документом в режиме разметки страницы и в режиме чтения.

ВЫ УЗНАЕТЕ:

- как применять стилевое форматирование для оформления документа;
- как в документе создать несколько разделов и отформатировать их;
- что такое колонтитулы и как их добавить в текстовый документ;
- как можно просмотреть структуру документа;
- как автоматически создать оглавление и указатель в текстовом документе;
- как создать текстовый документ на основе шаблона;
- как работать со сложными текстовыми документами.

ИЗУЧАЕМ

1. Как применять стилевое форматирование для оформления документа?

Внешний вид документа можно изменять, меняя значения параметров форматирования его символов, абзацев и страниц. Вы уже умеете изменять шрифт, цвет и размер символов, их начертание, выравнивание абзацев, отступы абзаца, междустрочный интервал и другие параметры форматирования символов и абзацев. Ускорить процесс форматирования текста в документе можно с помощью стилей.

Стилем называется совокупность параметров форматирования текстовых фрагментов, которую обозначают уникальным именем.

Стиль можно рассматривать как команду форматирования, созданную пользователем. Пользователь может выбрать значения параметров форматирования, которые нужно применить (например, шрифт и его размер, выравнивание, расстояние между символами, между абзацами, тип обрамления и т. п.), объединить их и задать уникальное имя этому набору значений параметров. Потом значения указанных параметров можно одновременно применить к выделенным текстовым фрагментам.

Для форматирования текста можно применить стили двух видов: стиль символа и стиль абзаца.

Стиль символа изменяет внешний вид отдельных символов, слов, фраз. Для настройки соответствующих параметров применяются инструменты для форматирования символов и параметры в окне *Шрифт*: шрифт и его размер, а также вид начертания, эффекты (зачёркнутый, верхний индекс, контур) и т. п.

Интересно

Стиль (в общем понимании) — характерный вид, разновидность чего-либо, выражающаяся в некоторых особенных признаках, свойствах художественного оформления.

Стиль абзаца позволяет изменить его внешний вид: шрифт и размер символов и другие атрибуты текста, а также междустрочный интервал, выравнивание текста, обрамление и значения других параметров, влияющих на форматирование всего абзаца.

Список *Стиль,* который в *Microsoft Word 2010* расположен на вкладке *Главная* и изображён на рисунке 9.1, *а* (в *LibreOffice Writer* — на панели инструментов *Форматирование*, рис. 9.1, *б*), содержит перечень доступных стилей при работе с текущим документом. При создании нового документа в этом списке автоматически отображаются встроенные стандартные стили. Затем, при создании пользователем новых стилей или изменении имеющихся, они добавляются к списку и в дальнейшем хранятся вместе с документом.

К стандартному набору стилей относятся такие стили: *Обычный* (*Основной текст*), значения параметров форматирования которого применяются к тексту, если не были указаны другие значения, *Заголовок 1, Заголовок 2, … , Заголовок 9,* с помощью которых можно формировать многоуровневую структуру больших сложных документов и т. п.

Чтобы применить стиль к тексту, достаточно выделить текстовый фрагмент, который необходимо отформатировать, и в списке *Стиль* выбрать нужный. Если названия нужных стилей не отображаются в списке *Стиль*, то можно открыть боковую панель *Стили* (*Стили и форматирование*), содержащую полный набор встроенных стилей, а также инструменты для внесения изменений в существующие стили и создания новых (рис. 9.2, *а*, *б*). Чтобы открыть панель *Стили*, в *Microsoft Word 2010* на вкладке *Главная* в группе *Стили* надо нажать кнопку , расположенную в правом нижнем углу группы. В *LibreOffice Writer* панель *Стили и форматирование* (рис. 9.2, *б*) появляется при выборе команды *Ещё стили* в конце списка стилей (рис. 9.1, *б*).

Разные стили в архитектуре

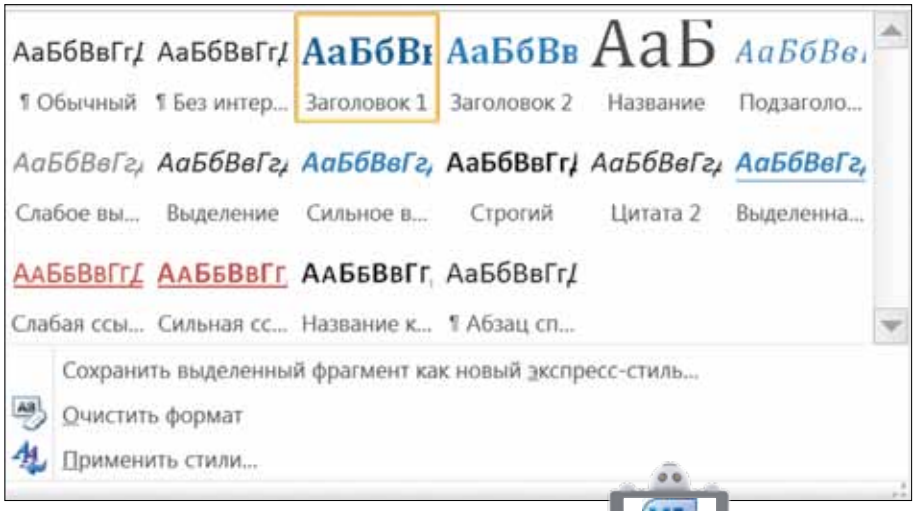


Рис. 9.1, *а*

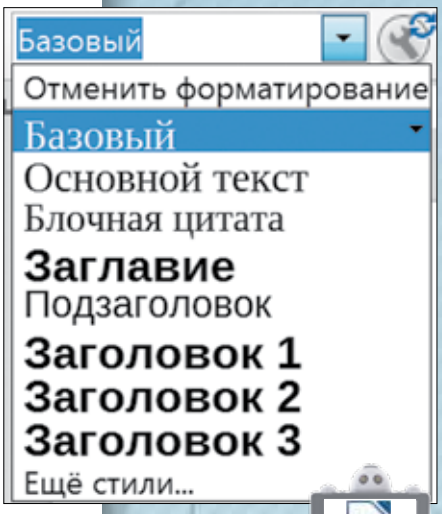


Рис. 9.1, *б*

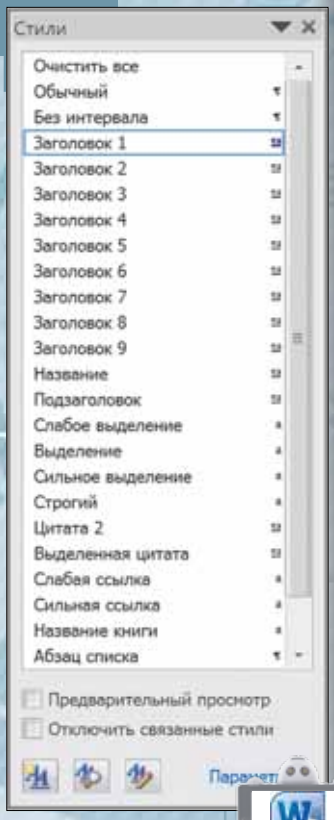


Рис. 9.2, *а*

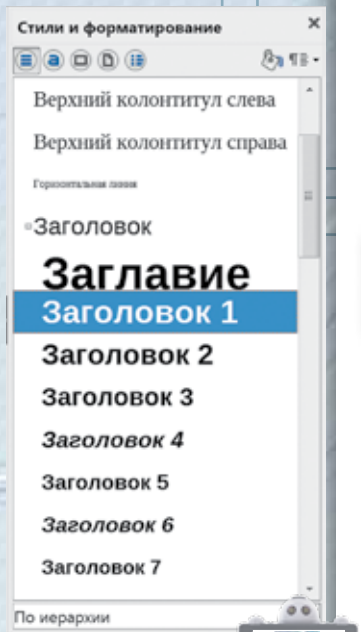


Рис. 9.2, *б*

ДЕЙСТВУЕМ

Упражнение 1. Форматирование документа с использованием стилей *Заголовки*.

Задание. В документе *Корреспонденция*, находящемся в папке *Текстовый процессор*, примените к названию документа стиль *Заголовок 1*, к подзаголовкам — стиль *Заголовок 3*.

1. В папке *Текстовый процессор* откройте документ *Корреспонденция*.
2. Выделите абзац, содержащий название документа. На вкладке *Главная* (на панели инструментов *Форматирование*) откройте список *Стиль* и выберите стиль *Заголовок 1*.
3. Аналогично примените к подзаголовкам в документе стиль *Заголовок 3*. Если название стиля *Заголовок 3* не отображается в списке *Стиль*, откройте область *Стили* (*Стили и форматирование*) (рис. 9.2, *а*, *б*).
4. Найдите в основном тексте слова в кавычках. Последовательно выделяйте каждое из таких слов или словосочетаний и примените к ним стиль *Выделение*.
5. Сохраните результаты в файле с тем же именем в папке *Тексты* своей структуры папок.

2. Как в документе создать несколько разделов и отформатировать их?

По умолчанию текстовый документ состоит из одного раздела. Все страницы одного раздела всегда имеют одинаковые значения параметров форматирования. Однако часто приходится работать с большими документами, состоящими из нескольких разделов. Например, электронный экземпляр книги, сборник правил или инструкций и т. п.

Как правило, новый раздел создаётся, если для разных страниц документа нужно задать разные значения параметров страницы или начинать с новой страницы определённую часть документа. Например, новый раздел в романе принято начинать с новой страницы. При разбивке текста на колонки автоматически создаются разделы, поскольку количество колонок — это один из параметров раздела.

> **Раздел** — часть документа, имеющая заданные значения параметров форматирования страницы или колонок.

Чтобы разбить документ на несколько разделов, следует вставить разрывы разделов, а затем задать значения параметров форматирования для каждого раздела.

В *Microsoft Word 2010* для создания нового раздела в документе нужно установить текстовый курсор в том месте документа, где нужно вставить разрыв раздела, и на вкладке *Разметка страницы* в группе *Параметры страницы* выбрать инструмент *Разрывы*. В раскрывающемся списке в области *Разрывы разделов* следует выбрать параметр, указывающий на начало нового раздела (рис. 9.3). При этом в документ вставляется метка ..... Разрыв раздела (со следующей страницы) ....., которую можно увидеть в режиме просмотра непечатаемых символов. Эта метка обозначает конец раздела и содержит сведения о таких параметрах его форматирования, как размеры полей, ориентация страницы, последовательность номеров страниц и т. п.

Рис. 9.3

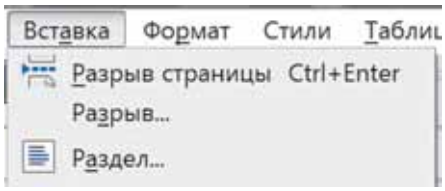


Рис. 9.4, *а*

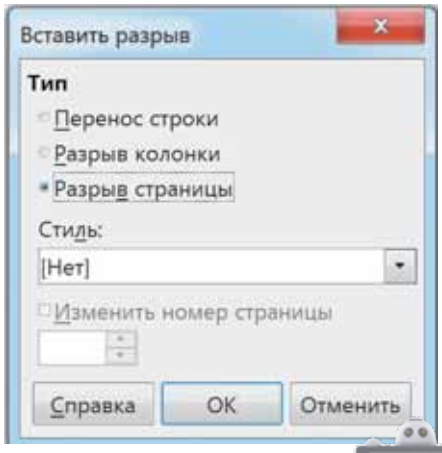


Рис. 9.4, *б*

Для создания нового раздела в *LibreOffice Writer* используется команда *Вставка/Раздел* (рис. 9.4, *а*). Если нужно, чтобы раздел начинался с новой страницы, выполняют команду *Вставка/Разрыв* и в окне *Вставить разрыв* выбирают тип *Разрыв страницы* (рис. 9.4, *б*).

В текстовом документе или его разделе можно изменять значения параметров форматирования страниц, в частности поля, ориентацию страницы, размер бумаги и т. п.

Поля страницы определяют расстояние от края листа бумаги до начала отображения текста.

По умолчанию документ печатается с одной стороны листа бумаги. В этом случае различают **верхнее**, **нижнее**, **левое** и **правое** поля (рис. 9.5). При подготовке журнальных и книжных изданий применяют двустороннюю печать. При этом вместо левого и правого полей используют понятия **внутренних** и **внешних** полей (рис. 9.6).

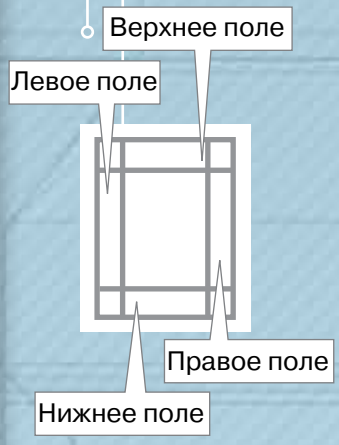


Рис. 9.5

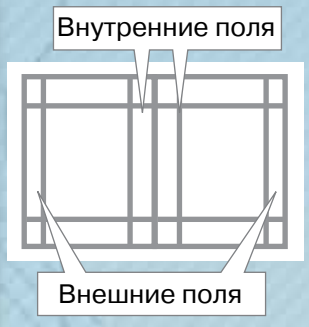


Рис. 9.6

Если предполагается сшивание напечатанного документа, то можно выбрать **положение переплёта** — слева или сверху, и задать значение дополнительного расстояния.

Подбирая **размер бумаги**, следует учитывать, на каком принтере документ будет напечатан. Так, лист формата A4 можно распечатать на любом современном принтере, тогда как формат бумаги A3 поддерживается только специальными принтерами.

Ориентация страницы может быть **книжная** — если страница располагается вертикально, или **альбомная** — если горизонтально (рис. 9.7).

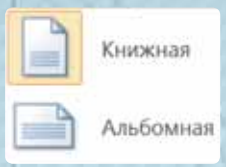


Рис. 9.7

Для изменения значений параметров форматирования страниц для определённого раздела в *Microsoft Word 2010* нужно установить текстовый курсор

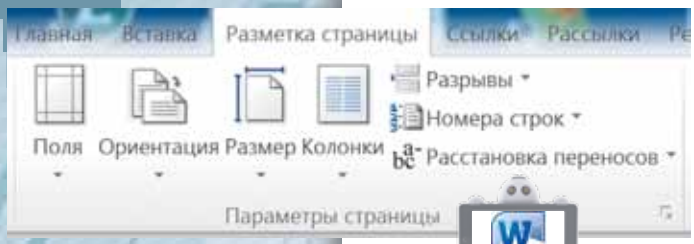


Рис. 9.8

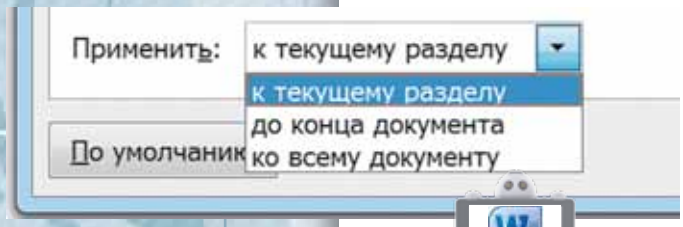


Рис. 9.9

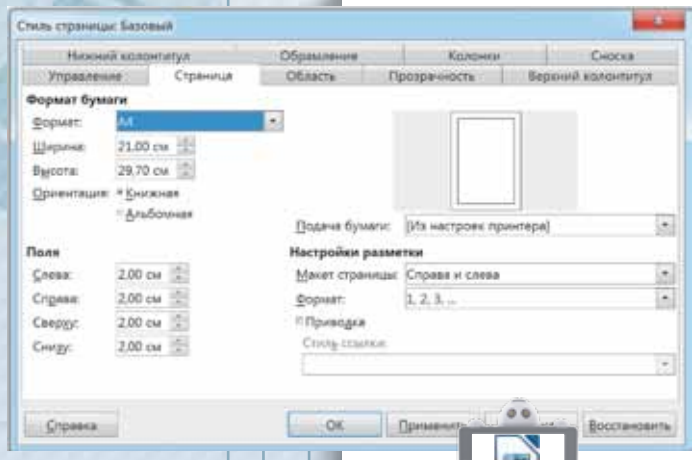


Рис. 9.10

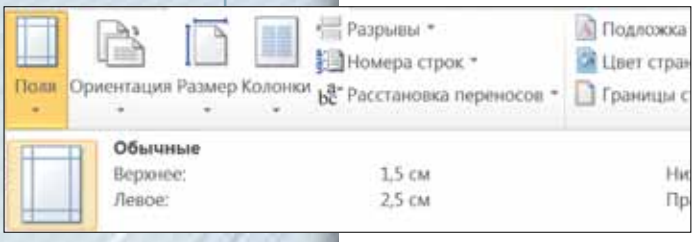


Рис. 9.11

в пределах этого раздела, на вкладке *Разметка страницы* в области *Параметры страницы* выбрать соответствующие инструменты (рис. 9.8) или нажать кнопку для вызова окна *Параметры страницы*, на соответствующих вкладках которого можно указать нужные значения параметров. Чтобы указанные изменения влияли не на весь документ, а только на выбранный раздел, нужно на любой из вкладок *Поля*, *Размер бумаги*, *Источник бумаги*, где изменялись значения параметров, в списке *Применить к* выбрать значение *к текущему разделу* (рис. 9.9).

Просмотреть и изменить значения параметров форматирования страницы в *LibreOffice Writer* можно в окне *Стиль страницы* (рис. 9.10), которое вызывают с помощью команды *Формат/Страница*. Размеры полей, ориентацию страницы и размер бумаги можно задавать на вкладке *Страница* этого окна.

ДЕЙСТВУЕМ

Упражнение 2. Создание разделов в документе.

Задание. Создайте в документе *Корреспонденция*, находящемся в папке *Тексты* своей структуры папок, разделы так, чтобы каждый пункт документа содержался в отдельном разделе и начинался с новой страницы. Установите ориентацию страницы для первого раздела — альбомную, а для всех разделов документа значения размеров полей: левое — 2,5 см, остальные — 1,5 см.

1. В папке *Тексты* своей структуры папок откройте документ *Корреспонденция*.
2. Установите значения параметров полей страниц для всего документа: левое — 2,5 см, остальные — 1,5 см, применив шаблон полей — *Обычный* (рис. 9.11).
3. Установите текстовый курсор в начале абзаца, содержащего подзаголовок *1. О письмах вообще*. На вкладке *Разметка страницы* в группе *Параметры страницы* выберите инструмент *Разрывы*. В раскрывающемся списке в области *Разрывы разделов* выберите *Следующая страница*.
4. Действуя аналогично, вставьте разделы в каждый пункт документа.
5. Установите текстовый курсор в пределах первого раздела. На вкладке *Разметка страницы* в области *Параметры страницы* выберите инструмент *Ориентация* и задайте альбомную ориентацию страницы.
6. Просмотрите документ. Сделайте выводы, изменилась ли ориентация страницы на альбомную для всех страниц документа.
7. Сохраните результаты в файле с тем же именем.

Рис. 9.12

3. Что такое колонтитулы и как их добавить в текстовый документ?

Важными параметрами страницы являются наличие и настройка колонтитулов, в которых можно указывать название документа или отдельного раздела, дату создания, фамилию автора, номер страницы, адреса интернет-ресурсов и т. п. Колонтитулы часто можно увидеть в книгах и журналах. Например, в книгах на чётных страницах может быть указан автор книги, а на нечётных — её название. В журналах на одном из колонтитулов обычно указывают название, номер журнала и год издания. Номера страниц, которые вы уже умеете добавлять к документу, размещаются именно в области верхнего или нижнего колонтитула документа (рис. 9.12).

Интересно

Название **колонтитул** образуется из двух слов: франц. *colonne* — столбец и лат. *titulus* — надпись, заголовок.

Колонтитул — объект, размещаемый над текстом (верхний колонтитул) или под текстом (нижний колонтитул) каждой страницы книги, газеты, журнала, документа.

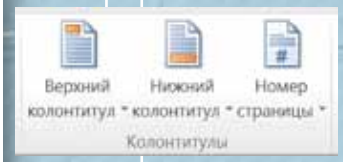


Рис. 9.13

Колонтитулы могут содержать текст и некоторые другие объекты: рисунки, линии, другие фигуры и т. п.

Если в документе есть несколько разделов, то колонтитулы могут быть одинаковыми для всех разделов или разными для каждого раздела.

В *Microsoft Word 2010* вводить и редактировать содержимое колонтитулов можно в отдельных областях в верхней или нижней части страницы. Чтобы перейти к области колонтитула, используют инструменты на вкладке *Вставка* в группе *Колонтитулы* (рис. 9.13). С помощью инструмента *Номер страницы* можно добавить и настроить в области колонтитула номера страницы. Особенно это удобно, когда в колонтитуле нет других объектов.

Список, раскрывающийся при выборе инструментов *Верхний колонтитул* или *Нижний колонтитул*, содержит библиотеку встроенных стилей колонтитулов (рис. 9.14), из которых можно выбрать наиболее подходящий по оформлению или наличию необходимых объектов.

После выбора одного из встроенных стилей колонтитулов текстовый курсор автоматически устанавливается в крайней левой части области колонтитула. Для размещения объектов по центру колонтитула или в правой его части используют клавишу *Tab,* позволяющую быстро перевести текстовый курсор в нужное место. При работе с колонтитулами текст документа отображается тусклым цветом и на ленте в области *Работа с колонтитулами* появляется вкладка *Конструктор* с инструментами для работы с колонтитулами (рис. 9.15).

Чтобы завершить создание или редактирование колонтитулов и вернуться к содержимому документа, нужно на вкладке

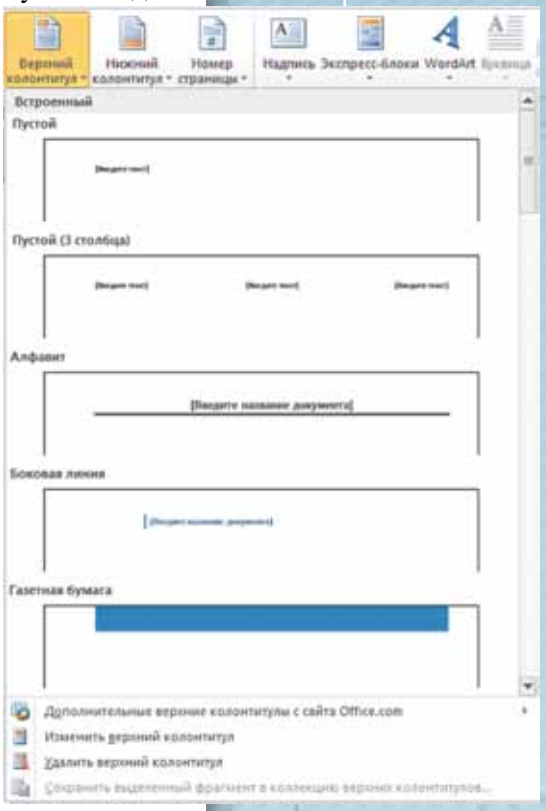


Рис. 9.14

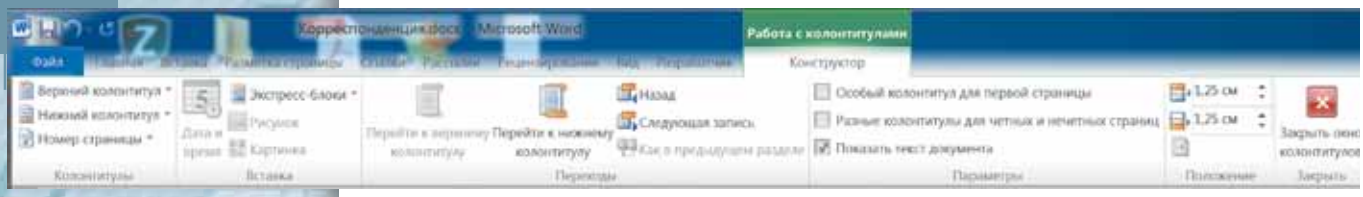


Рис. 9.15

Конструктор нажать кнопку *Закрыть окно колонтитулов* или дважды щёлкнуть мышью вне области колонтитулов.

Если документ состоит из нескольких разделов, то по умолчанию включён режим *Как в предыдущем разделе*, и колонтитулы всех разделов будут одинаковыми. Когда необходимо создать разные колонтитулы в каждом разделе, то после создания колонтитула первого раздела следует перейти к колонтитулу следующего раздела и на вкладке *Конструктор* выбрать инструмент *Как в предыдущем разделе* для выключения этого режима. После этого можно создавать новый колонтитул в выбранном разделе.

По умолчанию все страницы документа нумеруются последовательно. В пределах разделов можно настроить и другие параметры нумерации: указать, с какого номера начинать нумеровать страницы раздела, добавить к номеру страницы номер раздела, например, 1-1, 1-2, 1-3 и 2-1, 2-2 и т. п.

В *LibreOffice Writer* добавить колонтитулы в документ можно с помощью команд *Верхний колонтитул* и *Нижний колонтитул* из меню *Вставка*. При этом в тексте в пределах колонтитула применяется стандартный стиль колонтитулов.

ДЕЙСТВУЕМ

Упражнение 3. Создание колонтитула в документе.

Задание. В документ *Корреспонденция*, находящийся в папке *Тексты* своей структуры папок, вставьте нижний колонтитул, в котором будет отображаться название документа, текущая дата и номер страницы.

1. В папке *Тексты* своей структуры папок откройте документ *Корреспонденция*.
2. На вкладке *Вставка* в группе *Колонтитулы* выберите инструмент *Нижний колонтитул* (рис. 9.13) и шаблон *Алфавит*.
3. В левой части нижнего колонтитула введите с клавиатуры название документа *Корреспонденция*. На вкладке *Конструктор* в области *Работа с колонтитулами* (рис. 9.15) выберите инструмент *Дата и время*, в окне *Дата и время* выберите такой формат даты, чтобы месяц был представлен числом, например 3.11.2016, и нажмите кнопку *OK*.
4. Нажмите клавишу *Tab*, чтобы перейти к правой границе области колонтитула. На вкладке *Конструктор* в области *Работа с колонтитулами* выберите инструмент *Номер страницы*.
5. Выполните задержку мыши над инструментами вкладки *Конструктор* в области *Работа с колонтитулами* и определите их назначение.
6. На вкладке *Конструктор* выберите инструмент *Закрыть окно колонтитулов*.
7. Убедитесь, что созданный колонтитул повторяется на всех страницах документа.
8. Сохраните результаты в файле с тем же именем.

4. Как можно просмотреть структуру документа?

Заголовки разделов и подразделов документа определяют его структуру. В текстовых процессорах есть средства, с помощью которых можно просматривать структуру документа и использовать её для быстрого перемещения в нужное место документа. Чтобы воспользоваться такими средствами, необходимо, чтобы для абзацев, содержащих такие заголовки, значение параметра *Уровень* в окне *Абзац* отличалось от *Основной текст — Уровень 1*, *Уровень 2* и т. п. (рис. 9.16). Во встроенных стилях заголовков номер уровня соответствует номеру в названии стиля: *Заголовок 1* имеет *Уровень 1, Заголовок 2 — Уровень 2* и т. д. Поэтому для использования средств работы со структурой можно применить к заголовкам разделов и подразделов соответствующие стандартные стили заголовков.

Просмотреть структуру документа и переместиться в место документа, где начинается подраздел с выбранным заголовком, в *Microsoft Word 2010* можно с помощью панели *Навигация* (рис. 9.17), которая появляется на экране при включении на вкладке *Вид* в группе *Показать* флажка *Область навигации* (рис. 9.18).

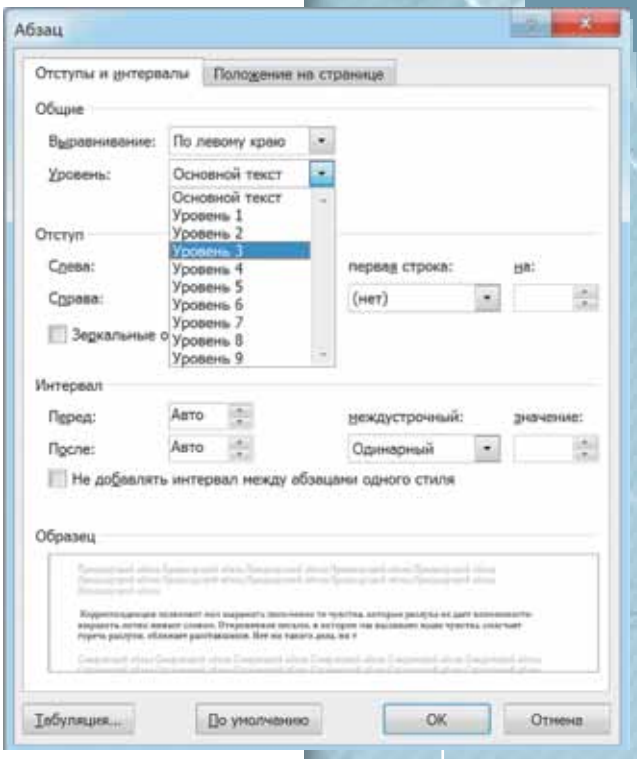


Рис. 9.16

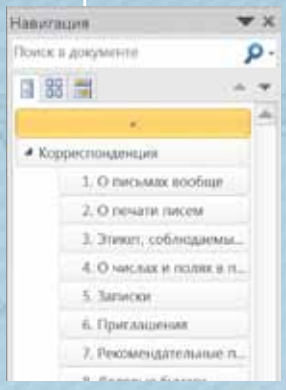


Рис. 9.17

Работать с большими документами в *Microsoft Word 2010* можно также в *Режиме структуры*, где можно свернуть документ, отображая только основные заголовки, перемещать или копировать части текста с помощью заголовков и т. п. В этом режиме текст представлен без учёта параметров форматирования абзацев. Все заголовки и обычный текст отображены с отступами, показывающими уровень этого текста в общей структуре документа (рис. 9.19). Слева от заголовков отображаются значки  , щелчок по которым позволяет выделить раздел, соответствующий выбранному заголовку, а двойной щелчок — свернуть или развернуть отображение подзаголовков выбранного заголовка.

Для перехода в режим структуры нужно на вкладке *Вид* в группе *Режимы просмотра документа* выбрать инструмент *Структура* или в правой части строки состояния нажать кнопку *Структура*  .

При работе в режиме структуры открывается вкладка *Структура* (рис. 9.20), содержащая в группе *Работа со структурой* инструменты, с помощью которых можно выбрать степень детализации. Например, отображать все заголовки или только заголовки от первого до третьего уровня, повысить или снизить уровень для определённых заголовков, перемещать подразделы в тексте и т. п.

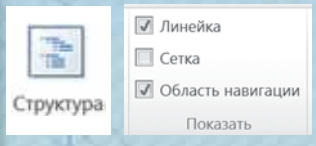


Рис. 9.18

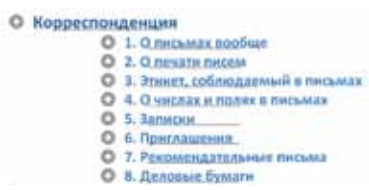


Рис. 9.19

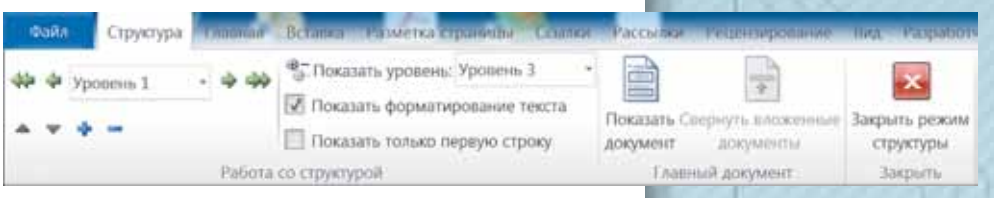


Рис. 9.20

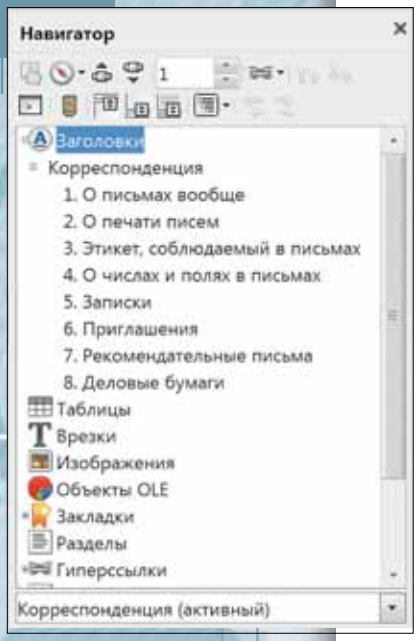


Рис. 9.21

Рис. 9.22

Для возврата в режим разметки документа, нужно на вкладке *Структура* выбрать инструмент *Закрыть режим структуры*.

В *LibreOffice Writer* заголовки и другие объекты документа можно просмотреть и использовать для перемещения в документе на панели *Навигатор* (рис. 9.21), которая открывается с помощью команды меню *Просмотр/Навигатор* или кнопки *Навигатор*  на боковой панели, расположенной в правой части окна (рис. 9.22). Заголовки более низких уровней могут быть скрыты. Для их отображения нужно щёлкнуть на значке ⊞ слева от заголовка первого уровня.

Упражнение 4. Просмотр документа в *Режиме структуры*.

Задание. Просмотрите документ *Повесть-сказка* в режиме отображения структуры. Не раскрывая на экране весь текст произведения, определите, из скольких разделов состоит повесть.

1. В папке *Текстовый процессор* откройте документ *Повесть-сказка*.
2. Перейдите в режим *Структура документа*.
3. Выберите второй уровень детализации структуры. Воспользуйтесь инструментом 

на вкладке *Структура* текстового процессора *Microsoft Word 2010* или значком ⊞ окна *Навигатор* в *LibreOffice Writer*.
4. Сосчитайте количество разделов документа. Выясните, как будет выглядеть документ при выборе третьего уровня структуры.
5. Сохраните изменения в файл *План повести* в папку *Тексты* своей структуры папок. Закройте окно текущего документа.
6. Откройте сохранённый файл. Проверьте, открылся ли режим отображения структуры после сохранения файла. Сделайте вывод. Закройте окно текстового процессора.

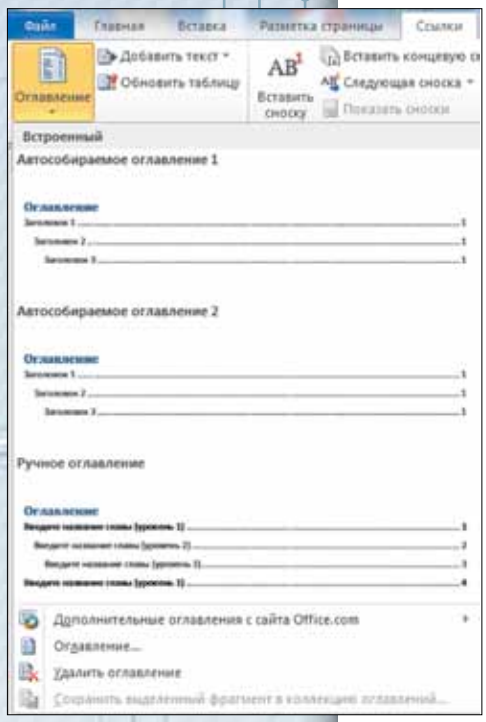


Рис. 9.23

5. Как автоматически создать оглавление и указатель в текстовом документе?

В больших документах, как правило, создают оглавление и другие средства для поиска нужного фрагмента в тексте. **Оглавление** — это список заголовков документа с указанием номеров страниц, на которых начинаются разделы с этими заголовками. Оглавление обычно размещают в начале документа. В конце больших документов, содержащих много новых терминов, создают **указатель** — словарь из новых терминов документа, расположенных в алфавитном порядке с указанием номера страницы, где этот термин встречается

в документе. Например, оглавление и указатель есть в учебнике по информатике, который вы держите в руках.

Текстовые процессоры позволяют автоматически создавать оглавление и указатель, которые можно также использовать для перемещения по документу, поскольку каждый элемент такого объекта является **ссылкой**. Например, если щёлкнуть мышью при нажатой клавише *Ctrl* на любом заголовке в оглавлении, текстовый курсор переместится в начало фрагмента документа с этим заголовком.

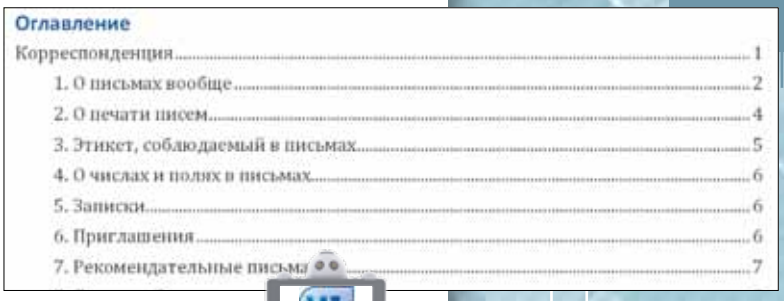


Рис. 9.24

Оглавление можно создать автоматически, при этом в него будут включены абзацы, у которых значение параметра *Уровень* отличается от *Основной текст* или в которых используются стандартные стили *Заголовок 1, Заголовок 2* и т. п. Поэтому, прежде чем добавить оглавление, необходимо сформировать структуру документа и установить текстовый курсор в то место в документе, где его нужно вставить.

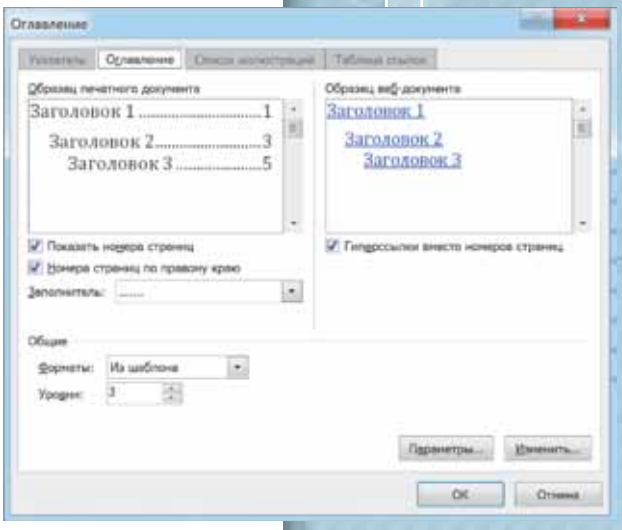


Рис. 9.25

Для добавления оглавления в *Microsoft Word 2010*, необходимо на вкладке *Ссылки* выбрать инструмент *Оглавление* (рис. 9.23).

Нужный стиль можно выбрать в раскрывающемся списке. При этом в документе автоматически будут найдены все абзацы со стилями заголовков, и в документе будет сформировано оглавление (рис. 9.24). Если необходимо задать особые настройки оглавления, следует выбрать команду *Оглавление...* (рис. 9.23). При этом открывается окно *Оглавление* (рис. 9.25), где можно выбрать режим вставки номеров страниц, с которых начинается каждый из разделов и подразделов, выбрать формат оформления, указать количество уровней заголовков для включения в оглавление и т. п.

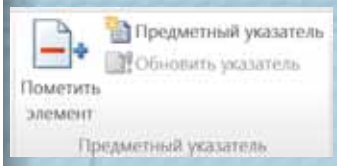


Рис. 9.26

Для создания указателя в *Microsoft Word 2010* используются инструменты из группы *Предметный указатель* на вкладке *Ссылки* (рис. 9.26).

Сначала в документе необходимо последовательно выделить термины, которые следует включить в указатель, и выбрать для каждого из них инструмент *Пометить элемент*. Когда все элементы отмечены, следует установить текстовый курсор в то место документа, где должен быть создан указатель, например в конце документа, и выбрать инструмент *Предметный указатель*. В окне *Указатель* (рис. 9.27), открывающемся при этом, можно изменить количество столбцов (колонок) для размещения терминов в указателе и другие параметры.

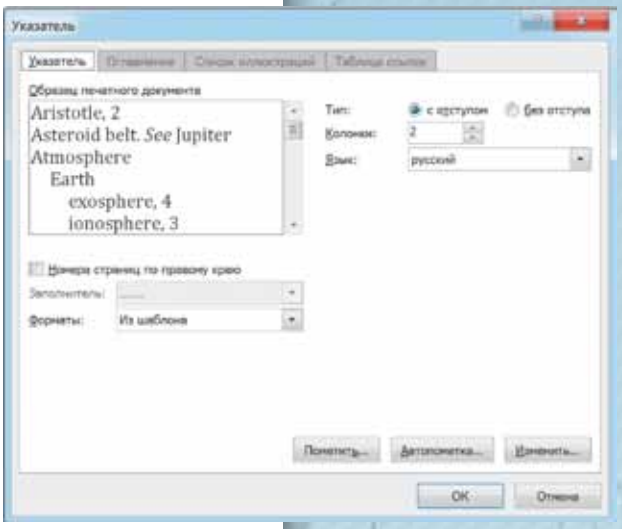


Рис. 9.27

Создание оглавления и указателя в *LibreOffice Writer* происходит аналогично. Для этого

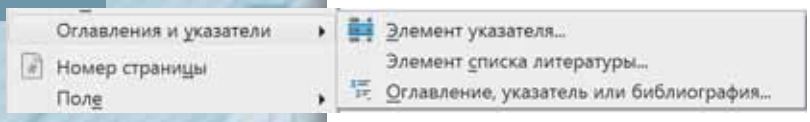


Рис. 9.28

используют команду меню *Вставка/Оглавления и указатели* (рис. 9.28).

Настройку параметров выполняют в окне *Вставить Оглавление/Указатель*.

ДЕЙСТВУЕМ

Упражнение 5. Создание оглавления документа.

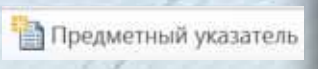
Задание. Для документа *Корреспонденция*, находящегося в папке *Тексты* своей структуры папок, создайте оглавление с помощью специальных инструментов.

1. В папке *Тексты* своей структуры папок откройте документ *Корреспонденция*. В этом документе к заголовкам разделов и подразделов в упражнении 1 вы применили стили *Заголовок 1* и *Заголовок 3*.
2. В начале документа вставьте новый абзац и установите в нём текстовый курсор.
3. На вкладке *Ссылки* выберите инструмент *Оглавление* и в раскрывающемся списке выберите один из стилей оформления оглавления (рис. 9.23).
5. Создайте новый раздел, чтобы основной текст документа начинался со следующей страницы после оглавления. Для этого на вкладке *Разметка страницы* в группе *Параметры страницы* выберите инструмент *Разрывы*. В раскрывающемся списке в области *Разрывы разделов* выберите *Следующая страница*.
6. Вернитесь к оглавлению. Нажмите на клавиатуре клавишу *Ctrl* и щёлкните на названии заголовка *Этикет, соблюдаемый в письмах* (рис. 9.24). Что при этом происходит?
7. Сохраните документ с тем же именем.

Упражнение 6. Создание алфавитного указателя.

Задание. Для документа *Корреспонденция*, находящегося в папке *Тексты* своей структуры папок, автоматически создайте предметный указатель терминов, используемых в документе.

1. В папке *Тексты* своей структуры папок откройте документ *Корреспонденция*.
2. Найдите в тексте документа следующие основные термины: *корреспонденция, письма, обращение, вступление, рассказ, вывод, постскриптум, конверты, этикет, записки, приглашение, рекомендательные письма, деловые бумаги*.
3. Последовательно выделяйте каждый из терминов и выбирайте инструмент *Пометить элемент* на вкладке *Ссылки* в группе *Указатель*.
4. Установите текстовый курсор в новом абзаце в конце документа и выберите инструмент *Предметный указатель* на вкладке *Ссылки*.
5. В окне *Указатель* просмотрите свойства, установленные по умолчанию (рис. 9.27), и нажмите кнопку *OK*.
6. Щёлкните мышью на слове *этикет* в созданном указателе. Убедитесь, что на экране отобразился фрагмент документа, содержащий этот термин. Сделайте вывод об использовании элементов указателя.
7. Сохраните документ с тем же именем.



6. Как создать текстовый документ на основе шаблона?

Для быстрого создания типовых документов — писем, визиток, резюме, отчётов и т. п. — используют шаблоны.

Шаблон документа — это текстовый документ, содержащий «общие» элементы для разных документов данного типа. Шаблон используется в качестве образца для создания новых документов определённого типа.

Как правило, в шаблоне определены стили, используемые в этих документах; шаблоны также могут содержать колонтитулы, любой готовый текст, изображения и т. п.

При создании нового документа по шаблону пользователь начинает не «с чистого листа», а с копии шаблона. Например, если существует готовый шаблон для приказов руководства компании, то при создании нового приказа его заголовок, изображение логотипа и т. п. уже размещены на своих местах, нужно только добавить номер и текст приказа.

Шаблоны часто используют для того, чтобы установить единый стандарт для типовых документов в организации.

Шаблон — внешне это обычный текстовый документ, но у него другое расширение. Например, в *Microsoft Word* файлы документов имеют расширение *doc* или *docx*, а шаблонов — *dot* или *dotx*. В *LibreOffice Writer* у файлов шаблонов расширение *ott*.

При создании документа на основе шаблона открывается не сам файл шаблона, а создаётся новый документ, содержащий все объекты файла шаблона. При внесении изменений в такой документ файл шаблона остаётся неизменным, поэтому его можно использовать многократно. Текстовые процессоры также позволяют открывать и редактировать шаблоны аналогично редактированию документов. Можно преобразовать документ в шаблон. Основное отличие между документами и шаблонами заключается в их использовании.

Существует два основных типа шаблонов: глобальные и шаблоны документов. Любой документ создаётся на основе шаблона. По умолчанию новые документы создаются на основе глобального шаблона *Обычный* (*Normal*), содержащего параметры форматирования, доступные для всех документов. Шаблоны документов, например шаблоны записки или резюме, содержат параметры форматирования, доступные только для документов, созданных с помощью этого шаблона.

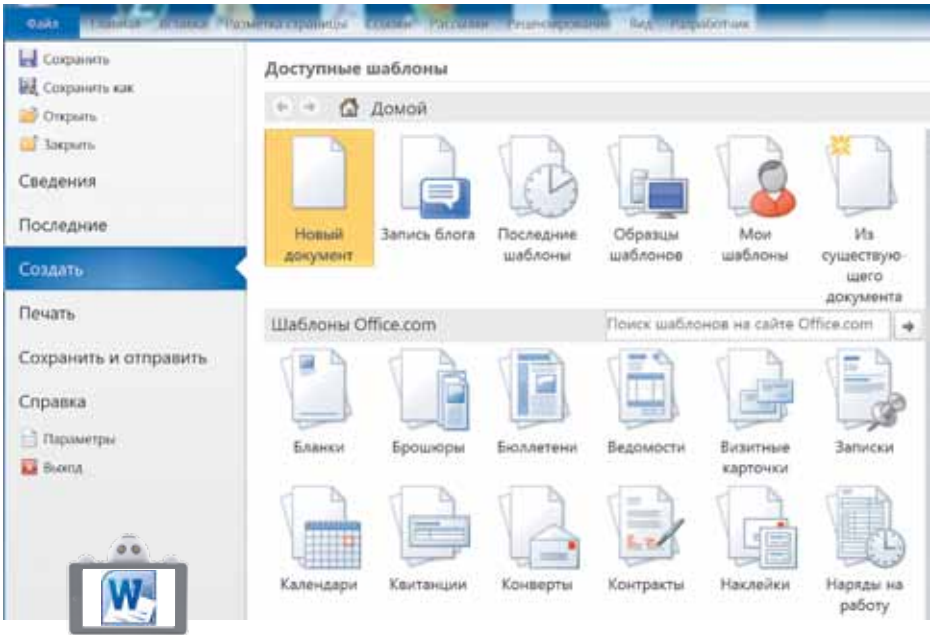


Рис. 9.29

Интересно

В расширении файлов шаблонов одна из букв расширения текстового файла заменяется буквой *t* — от англ. *template* — шаблон.

Интересно

Если необходимо создать набор документов, имеющих похожее форматирование, желательно использовать шаблоны из одной группы: *Обычные*, *Изысканные* или *Современные*. Так, если для создания резюме выбрали шаблон *Изысканное резюме*, то для подготовки письма к нему стоит также использовать шаблон *Изысканное письмо*.

Рис. 9.30

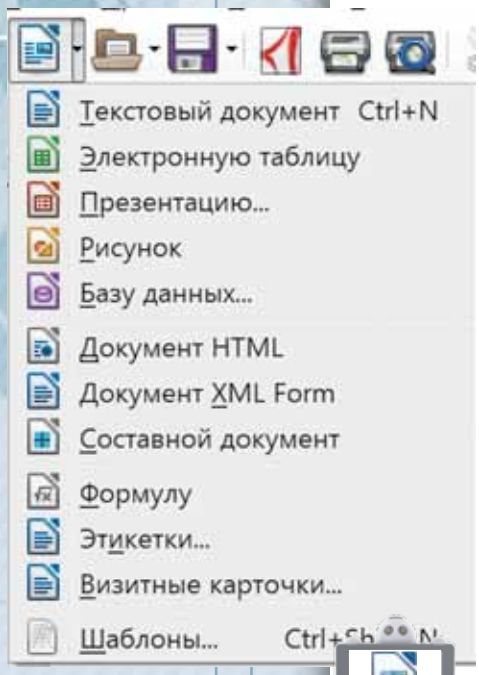


Рис. 9.31

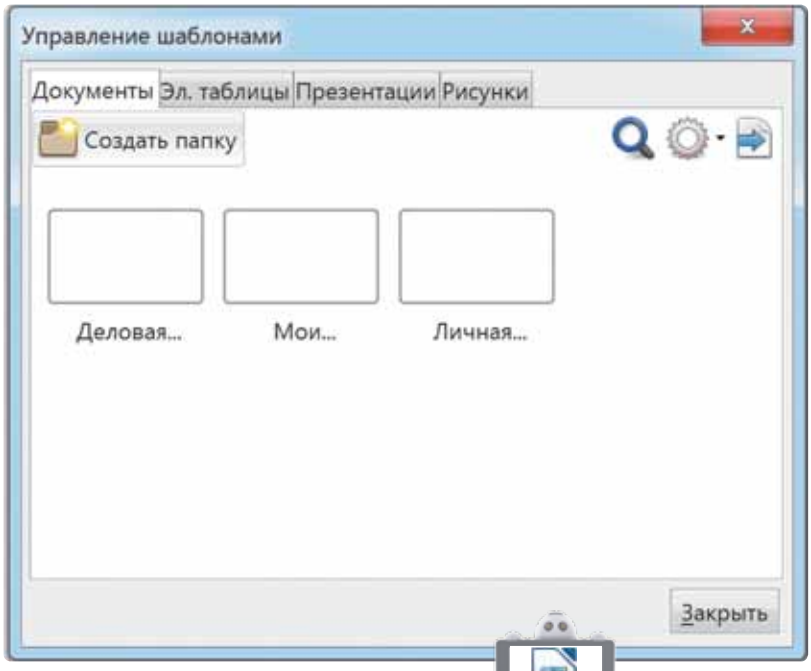


Рис. 9.32

Шаблоны создаются в помощь пользователю, их использование сэкономит время для подготовки документов. Разные текстовые процессоры имеют собственные наборы шаблонов.

Пакет программ *Microsoft Office* содержит большое количество различных шаблонов, некоторые из текстовых шаблонов можно открыть непосредственно в текстовом процессоре, другими — шаблонами *Office.com* — можно воспользоваться при наличии подключения к Интернету.

Для создания документа на основе шаблона документа в текстовом процессоре *Microsoft Word 2010* используют команду *Файл/Создать*, в области *Доступные шаблоны* следует выбрать *Образцы шаблонов* (рис. 9.29) и среди предложенных выбрать нужный шаблон документа. В правой части экрана нужно установить переключатель в положение *Документ* и нажать кнопку *Создать* (рис. 9.30). Далее необходимо ввести нужный текст в каждое поле с соответствующими подсказками такого документа.

Документ можно создавать и на основе нестандартного шаблона, для этого открывают шаблон как обычный документ, а затем работают с его содержимым.

В текстовом процессоре *LibreOffice Writer* инструмент *Создать* на панели инструментов *Стандартная* содержит раскрывающийся список для создания стандартных документов пакета *LibreOffice* (рис. 9.31). Из дополнительных шаблонов текстовых документов можно выбрать *Формулу*, *Этикетки* и *Визитные карточки*. Также можно создавать новые *Шаблоны* с использованием инструментов окна *Управление шаблонами* (рис. 9.32).

ДЕЙСТВУЕМ

Упражнение 7. Создание документа на основе шаблона.
Задание. Создайте новый документ на основе шаблона *Анкета.dotx* (*Анкета.ott*), содержащегося в папке *Текстовый процессор*.

1. Откройте папку *Текстовый процессор* и дважды щёлкните на значке шаблона *Анкета.dotx* (*Анкета.ott*).
2. Просмотрите содержимое нового документа. Определите, используя строку заголовка окна текстового процессора, сохранён ли новый документ и какое у него имя.
3. Используя предложенный шаблон, заполните анкету «Какой я?».
4. Сохраните документ с именем *Анкета.docx* (*Анкета.odt*) в папке *Тексты* своей структуры папок.

Упражнение 8. Создание документа с помощью встроенного шаблона.
Задание. Создайте файл *Письмо* с помощью соответствующего стандартного шаблона текстового процессора.

1. Загрузите текстовый процессор *Microsoft Word 2010.* Выполните команду *Файл/Создать,* в области *Доступные шаблоны* выберите *Образцы шаблонов.* Среди перечня шаблонов выберите *Современное письмо*, в правой части окна установите переключатель в положение *Документ* и нажмите кнопку *Создать*.
2. Введите своё имя и вместо текстовых фрагментов, записанных в квадратных скобках, например [*Введите адрес отправителя*], введите собственные данные.
3. Подготовьте текст письма с просьбой к авторам о предоставлении разрешения использовать их произведение, например, для исследовательского проекта.
4. Удалите предложенное изображение и вставьте на его место своё фото.
5. Сохраните документ с именем *Письмо* в папке *Тексты* своей структуры папок.

7. Как работать со сложными текстовыми документами?

Эффективная работа со сложным документом с большим количеством страниц, состоящим из нескольких разделов, предусматривает выполнение определённых действий:

1. Ввод текста и добавление объектов к документу.
2. Если необходимые текстовые фрагменты сохранены в другом файле, можно воспользоваться буфером обмена для их копирования и вставки в документ. Объединять текст нескольких файлов можно с помощью команды *Текст из файла* из списка инструмента *Объект*, расположенного на вкладке *Вставка* в группе *Текст* (рис. 9.33).
   Далее в окне *Вставка файла* выбирают нужный файл и нажимают кнопку *Вставить*.

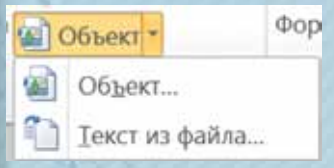


Рис. 9.33

3. Форматирование текста и объектов документа.
4. Форматирование заголовков разделов и пунктов с использованием встроенных стандартных стилей заголовков или пользовательских параметров форматирования, определяющих уровень абзаца.
5. Создание в документе необходимого количества разделов и настройка параметров страницы для каждого раздела отдельно или для всего документа.
6. Просмотр структуры документа с помощью средств навигации или в режиме структуры, при необходимости коррекция последовательности расположения разделов.
7. Добавление к документу средств ориентирования и навигации: оглавления, предметного указателя и т. п.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 9* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

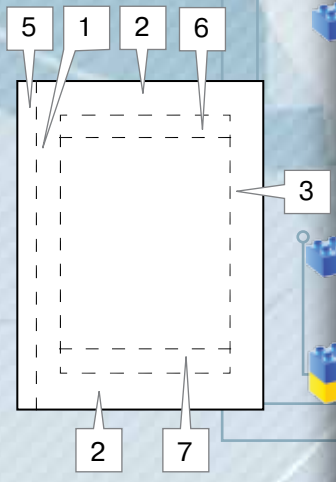


Рис. 9.34

1. Изобразите средствами текстового процессора модель документа, содержащего указанные элементы по образцу (рис. 9.34):
   - верхний колонтитул;
   - нижний колонтитул;
   - левое поле;
   - правое поле;
   - переплёт;
   - верхнее поле;
   - нижнее поле.

   Предложите в паре определить их размещение на модели.
2. Как внести некоторые изменения в любой имеющийся шаблон? Можно ли создать набор шаблонов для ученика в школе? Назовите пять примеров таких документов. Обсудите в парах. Создайте общий список.
3. Если создать собственный шаблон и сохранить его в стандартной папке *Шаблоны*, то будет ли такой шаблон отображаться в диалоговом окне *Шаблоны*, появляющемся на экране после выбора на панели задач ссылки *На моём компьютере*? Если будет, то в каком разделе, с каким названием? Обсудите в парах.
4. Для чего в печатные издания вставляют колонтитулы? Назовите три преимущества использования колонтитулов в печатных документах. Могут ли в одной книге содержаться разные колонтитулы? Обсудите в парах.
5. Обсудите возможные причины, когда в документе целесообразно использовать оглавление, а когда — режим структуры.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Создайте собственную визитку на основе соответствующего встроенного шаблона.
2. В папке *Текстовый процессор* откройте документ *Памятники Киева*. Воспользуйтесь панелью навигации и определите, сколько заголовков содержит документ. Просмотрите, какие стили имеют заголовки документа. Установите следующие параметры форматирования страницы: *левое поле* — 3 см, *правое поле* — 1,5 см.

3. В папке *Текстовый процессор* откройте документ *Памятники Киева*. Определите, какие стили имеют заголовки документа. Создайте оглавление в начале документа. Добавьте в документ номера страниц и нижний колонтитул, содержащий текст *Памятники Киева*.
4. В папке *Текстовый процессор* откройте документ *Структура делового письма*. Примените к заголовкам и подзаголовкам документа стили заголовков соответствующих уровней. Создайте оглавление в начале документа. Добавьте в конце документа материалы из файла *Дополнения* папки *Текстовый процессор*. Воспользуйтесь буфером обмена или командой *Текст из файла*.
5. В папке *Текстовый процессор* откройте документ *Памятники Киева*. Разбейте документ на разделы таким образом, чтобы сведения о каждом из памятников начинались с новой страницы. Создайте в документе колонтитулы, в которых на нечётных страницах будет отображаться название *Памятники Киева*, на чётных — название памятника, относящегося к определённому разделу. Чтобы иметь возможность в разных разделах создать разные колонтитулы, отключите режим *Как в предыдущем разделе*, для этого выберите соответствующий инструмент на вкладке *Конструктор* в области *Работа с колонтитулами*.
6. Создайте похвальную грамоту на основе шаблона *Грамота*, находящегося в папке *Текстовый процессор*. На его основе создайте грамоту для конкретного человека за успешное изучение текстового процессора.
7. Оформите реферат, который вы недавно готовили по одному из учебных предметов, по алгоритму работы со сложным документом. Предусмотрите использование стилей заголовков, разбивку документа на разделы, настройку колонтитулов и параметров страницы документа, автоматическое добавление оглавления в начало документа.

ИССЛЕДУЕМ

Определите, какие параметры на вкладке *Источник бумаги* в окне *Параметры страницы* в *Microsoft Word 2010* (на вкладках *Верхний колонтитул* и *Нижний колонтитул* в окне *Стиль страницы* в *LibreOffice Writer*) используются для настройки колонтитулов.

ПОЛЕЗНЫЕ ССЫЛКИ

Создание и обновление оглавления:
https://support.office.com/ru-ru/article/Создание-и-обновление-оглавления-eb275189-b93e-4559-8dd9-c279457bfd72

Создание и обновление предметного указателя:
https://support.office.com/ru-ru/article/Создание-и-обновление-предметного-указателя-cc502c71-a605-41fd-9a02-cda9d14bf073

10. ПРАКТИЧЕСКАЯ РАБОТА 5

СТРУКТУРА ДОКУМЕНТА. АВТОМАТИЗИРОВАННОЕ СОЗДАНИЕ ОГЛАВЛЕНИЯ И УКАЗАТЕЛЕЙ

ВСПОМНИТЕ

- Как с помощью стилей обозначить заголовки в большом документе;
- как просмотреть структуру текстового документа;
- как добавлять к текстовому документу колонтитулы, автоматизированное оглавление и предметный указатель;
- как форматировать параметры страницы в текстовом документе.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 5.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Правописание основы слова (*10 баллов*)

Определите, какие действия выполнялись при подготовке документа *Правописание основы слова*, содержащегося в папке *Текстовый процессор*:

- применение стилей заголовков;
- добавление колонтитулов;
- разбивка документа на разделы;
- установка для некоторых разделов разных параметров страницы;
- автоматизированное добавление оглавления документа.

Откройте панель навигации и просмотрите структуру документа. Определите, сколько уровней заголовков выделено в документе. Воспользуйтесь заголовками на панели *Навигация* для перемещения в начало каждого из разделов.

Просмотрите содержимое документа. Определите основные понятия, используемые в документе, и обозначьте их как элементы указателя. В конце документа создайте указатель.

Задание 2. Авиаконструктор Антонов (*10 баллов*)

Внесите изменения в документ *Антонов*, находящийся в папке *Текстовый процессор*: примените стили к заголовкам, создайте оглавление, разбейте документ на разделы так, чтобы оглавление было расположено на отдельной странице и каждый фрагмент текста, содержащий заголовок, начинался с новой страницы. Для раздела с оглавлением установите ориентацию страницы — *альбомная* и значения левого и правого полей — по *2 см*. Добавьте в документ колонтитул, содержащий ссылку на источник материала (ссылка указана в конце документа) и номера страниц. Шаблон и расположение колонтитула (верхний или нижний) выберите самостоятельно.

Задание 3. Произведения Леси Украинки (*14 баллов*)

По правилам создания сложных документов создайте текстовый документ *Произведения Леси Украинки*, объединяющий находящиеся в папке *Текстовый процессор\Леся Украинка* текстовые документы, содержащие поэмы и стихотворения. Примените к абзацам с названиями произведений стиль *Заголовок 2*. Добавьте в документ колонтитулы. Шаблон, содержимое и расположение (верхний или нижний) колонтитула выберите самостоятельно. В начале документа создайте автоматизированное оглавление.

РАЗДЕЛ 4

РАБОТА С ОБЪЕКТАМИ МУЛЬТИМЕДИА

11. РАБОТА С АУДИО- И ВИДЕОФАЙЛАМИ

ВСПОМНИТЕ:

- что понимают под мультимедиа;
- какие устройства используются для работы с объектами мультимедиа;
- с помощью каких программ можно просматривать видео и слушать аудиозаписи.

ВЫ УЗНАЕТЕ:

- какие существуют форматы файлов звукозаписи;
- какие форматы видеофайлов являются наиболее распространёнными;
- с помощью каких программ можно преобразовать звукозаписи и видеофайлы из одного формата в другой;
- какие сервисы предназначены для размещения видео- и аудиоматериалов в Интернете;
- как искать и просматривать видео на *YouTube;*
- какие программы используются для обработки аудиозаписей;
- с помощью каких программ можно создать собственный видеоклип;
- какие особые элементы интерфейса содержит программа *Киностудия.*

ИЗУЧАЕМ

1. Какие существуют форматы файлов звукозаписи?

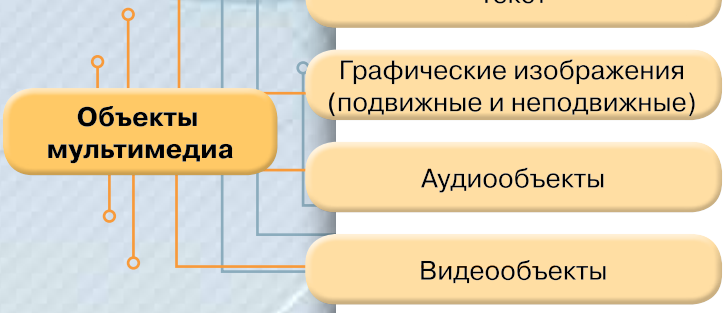


Рис. 11.1

Вы уже знаете, что объекты мультимедиа (рис. 11.1) хранятся в виде файлов с мультимедийными данными. Форматы аудио- и видеофайлов определяют их структуру и способы кодирования. Способы кодирования звуковых и видеоданных в разных форматах определяют качество звука или воспроизведения видео и степень сжатия данных, влияющую на объём файлов.

Наиболее распространены такие форматы звукозаписи (табл. 11.1):

Таблица 11.1

Аудиоформат	Полное название формата	Особенности формата
MIDI	***M****usical* ***I****nstrument* ***D****igital* ***I****nterface*	В отличие от других аудиоформатов, представляет не оцифрованный звук, а наборы команд (инструмент, проигрываемые ноты, значения параметров звука и т. п.), которые могут воспроизводиться по-разному, в зависимости от устройства воспроизведения. Позволяет обмениваться данными между музыкальными инструментами, синтезаторами и компьютерами
WAV	***Wav****eform audio format* (от англ. *wave* — волна)	Используется в операционной системе *Windows*. Аудиоформат — без использования сжатия. Точно передаёт звук, но занимает значительный объём на диске

Продолжение таблицы 11.1

Аудио-формат	Полное название формата	Особенности формата
MP3	***MP**EG Layer **3***	Запись музыки в этом формате происходит со сжатием объёма с почти незаметным для слуха ухудшением качества, но объём уменьшается в 10–12 раз по сравнению с оригинальным музыкальным форматом. Принцип сжатия данных напоминает графический формат JPEG — сжатие происходит за счёт исключения частот, которые не слышит человек
WMA	***W**indows **M**edia **A**udio*	Разработан компанией *Microsoft* как альтернатива формату MP3. Степень сжатия данных и качество звука почти аналогичны формату MP3. Новые версии формата, начиная с *Windows Media Audio 9.1*, предусматривают кодирование без потери качества, многоканальное кодирование объёмного звука и кодирование голоса
AAC	***A**dvanced **A**udio **C**oding*	При кодировании значительно уменьшается объём данных, необходимых для передачи высококачественного цифрового аудио. В этом формате происходит меньшая потеря качества, чем в MP3, при одинаковых объёмах данных

2. Какие форматы видеофайлов являются наиболее распространёнными?

Форматы видео представляют собой **медиаконтейнеры**, т. е. могут содержать данные разных типов, сжатые разными кодеками, и позволяют хранить аудио-, видео- и текстовые данные (в частности, субтитры) в одном файле. Медиаконтейнер не только предоставляет возможность хранения аудио- и видеозаписей, но и обеспечивает синхронизацию аудио- и видеопотоков при воспроизведении.

Видео может быть сохранено на разных носителях. Как правило, видео высокого качества имеет значительный объём. Например, объём музыкальной комедии «Сорочинская ярмарка», записанной на DVD, — 6,71 Гб.

Как и форматы звукозаписи, разные форматы видеофайлов предусматривают разные способы кодирования данных, определяющие качество видео, степень сжатия данных и объём файла. Некоторые форматы могут содержать **потоковое** видео, используемое для передачи данных через Интернет в режиме реального времени.

Существуют такие распространённые форматы видео (табл. 11.2):

Таблица 11.2

Видео-формат	Полное название формата	Особенности формата
AVI	***A**udio-**V**ideo **I**nterleaved*	Может содержать потоки четырёх типов: видео, аудио, MIDI, текст. Для сжатия аудио- и видеозаписей могут использоваться разные кодеки. Имеет некоторые ограничения, в частности, объём файла не может превышать 4 Гб. На смену этому формату создан формат WMV

Интересно

Кодек (от англ. *coder/decoder* — кодировщик/декодировщик) — устройство или программа, выполняет преобразование сигналов и использует при цифровой обработке видео и звуков для сжатия данных. Сжатие, как правило, происходит с потерей качества. Кодеки позволяют кодировать видеозаписи для передачи или хранения, а также раскодировать — для просмотра. Разные медиаконтейнеры могут поддерживать разные кодеки: DivX, XviD, MJPEG, VC-1 и т. п.

Интересно

Плагин (от англ. *plug-in* — подключать) — независимый программный модуль, подключаемый к основной программе и предназначенный для расширения или использования её возможностей.

Часто в виде плагина выполняется поддержка форматов файлов, например, для звуковых и видеопроигрывателей, программ обработки звука и графики и т. п. В веб-браузерах плагины используются для обеспечения отображения форматов данных, не имеющих встроенной поддержки браузером (например, *Adobe Flash* или *SVG*), для расширения возможностей согласно требованиям пользователя и т. п.

Продолжение таблицы 11.2

Видеоформат	Полное название формата	Особенности формата
MPEG	***M**otion **P**icture **E**xperts **G**roup*	Разработан *экспертной группой по движущемуся изображению* (*MPEG*). Были созданы такие алгоритмы сжатия данных: MPEG1, MPEG2 и MPEG4
MOV	*QuickTime **Mov**ie*	Один из первых видеоформатов, получивших широкое распространение. Степень сжатия — достаточно высокая
ASF	***A**dvanced **S**ystems **F**ormat* (ранее также *Advanced Streaming Format, Active Streaming Format*)	Является частью мультимедийного набора *Windows Media* для создания и распространения аудио- и видеофайлов. Может использоваться как для локального воспроизведения, так и для передачи и воспроизведения по компьютерным сетям, в частности Интернету. Особенностью формата является возможность воспроизведения непосредственно в момент загрузки по сети в режиме реального времени, напоминающая телевизионное вещание, т. е. *потоковое воспроизведение*. Обычно используется расширение файла *asf*, кроме того, файлы, содержащие звуковые записи, могут иметь расширение *wma*, а видеофайлы — *wmv*
WMV	***W**indows **M**edia **V**ideo*	Является частью мультимедийного набора *Windows Media*. Создан на основе формата AVI, но имеет дополнительные возможности, в частности, средства защиты от несанкционированного копирования. Используется для распространения фильмов и видеоклипов
3GP	***3**rd **G**eneration (mobile) **P**hone*	Формат для сохранения и просмотра видео на мобильных телефонах 3-го поколения. У видеозаписей в этом формате — небольшой объём по сравнению с другими форматами видео, однако достигается это за счёт ухудшения качества
RM	***R**eal **M**edia*	Разработан компанией *Realnetworks* для распространения видео через Интернет. Степень сжатия данных и качество видео достаточно высокие. Используется для распространения фильмов и трансляции так называемого «интернет-телевидения»
VOB	***V**ideo **Ob**ject*	Формат файлов, используемый для хранения DVD-видео. Создан на основе MPEG2, может содержать несколько потоков аудио, видео, субтитры, а также меню фильма. Используется для распространения фильмов на DVD
FLV	***Fl**ash **V**ideo*	Используется для передачи видео через Интернет, в частности, такими сервисами: *YouTube*, *Вконтакте*, *RuTube* и др. Файлы в этом формате можно просматривать в большинстве операционных систем, т. к. для этого используется проигрыватель *Adobe Flash Player*, который распространяется в виде *плагина* для разных браузеров и разных операционных систем

3. С помощью каких программ можно преобразовать звукозаписи и видеофайлы из одного формата в другой?

На практике часто возникают ситуации, когда нужно преобразовать аудио- или видеофайлы в другой формат. Например, видеозаписи, снятые на мобильный телефон в формате 3GP, для последующей обработки могут быть преобразованы в формат AVI или WMV.

Для преобразования файлов из одного формата в другой используются специальные программы — **конвертеры**, их выбор зависит от исходного формата файла и формата, в который его нужно преобразовать.

Конвертер — программа для преобразования в файле данных из одного формата в другой. Изменения и потери данных, возникающие при преобразовании, зависят от форматов начального и конечного файлов и от используемой программы преобразования.

Интересно

Конвертер — от англ. *convertere* — преобразовывать.

Существуют различные программы, предназначенные для преобразования аудио- или видеофайлов, некоторые из них бесплатные. Такие программы можно загрузить из Интернета (рис. 11.2).

После выбора одной из ссылок на программы-конвертеры, как правило, отображается описание программы и предлагается ссылка для загрузки файла, с помощью которого происходит установка программы на компьютер.

Аннотация позволяет определить, какие конвертеры можно загрузить на указанных страницах, а также являются ли такие программы бесплатными.

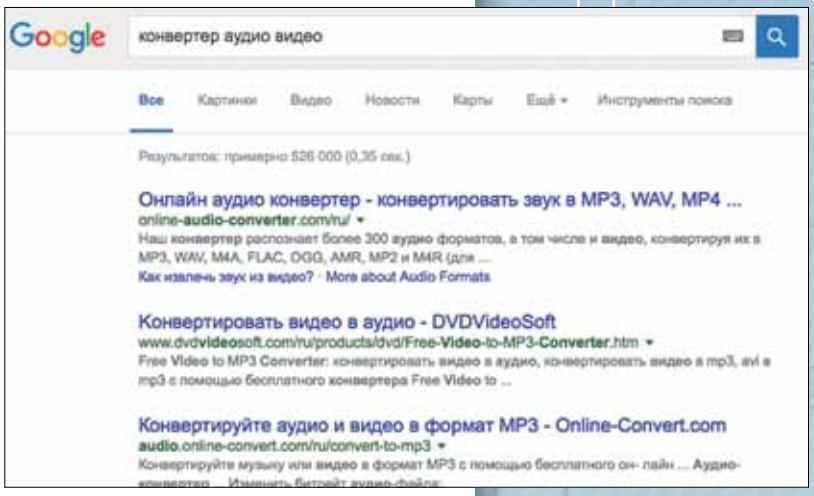


Рис. 11.2

4. Какие сервисы предназначены для размещения видео- и аудиоматериалов в Интернете?

Сегодня на различных сервисах можно размещать, просматривать и прослушивать видео- и аудиоматериалы. Некоторые из них бесплатные, другие предлагают различные тарифные планы. Большинство сервисов для размещения материалов предусматривают предварительную регистрацию пользователя.

YouTube является крупнейшим в мире видеосервисом для просмотра и размещения в Интернете видео, созданного пользователями со всего мира. Просматривать видео на *YouTube* может любой. На портале *https://www.youtube.com* есть система поиска, с помощью которой можно быстро найти нужные видеоматериалы.

Видео на сайте *YouTube* пользователи размещают самостоятельно, также они могут оставлять комментарии и оценки. Правила сообщества и сообщения на сайте свидетельствуют о том, что пользователи должны иметь авторские права или разрешение владельцев прав на размещение любого видеоматериала. Размещать видео, добавлять комментарии и оценки могут только зарегистрированные пользователи.

Интересно

Компания *YouTube* основана в феврале 2005 г. Самое первое видео на *YouTube* было размещено 23 апреля 2005 г. В октябре 2006 г. *Google* приобрёл *YouTube*, но по соглашению за *YouTube* остались торговая марка и особенности бренда.

Список ссылок на видео- и аудиосервисы

Существуют также другие сервисы для размещения и просмотра видео, в частности *https://vimeo.com* и *http://rutube.ru*.

На некоторых веб-порталах есть встроенные сервисы для размещения аудио- и видеоматериалов. Например, портал *http://meta.ua* содержит сервисы для размещения видео и аудио — *http://video.meta.ua* и *http://audio.meta.ua*, а также для поиска и прослушивания радио онлайн — *http://radio.meta.ua*.

Примерами сервисов для размещения и поиска аудиозаписей являются *https://podfm.ru/*, *http://www.audiomicro.com*, *http://boomp3.com*, *https://soundcloud.com*, *http://freemusicarchive.org*.

Аудиозаписи, содержанием которых являются не музыка, а радиопередачи, аудиоспектакли, интервью, лекции и т. п., могут быть размещены в виде подкастов. **Подкастом** называется отдельный аудиофайл либо серия файлов, которые публикуются по одному адресу в Интернете и регулярно обновляются.

Множество сайтов позволяют слушать аудиозаписи онлайн. Это могут быть любимые песни, аудиокниги, радиопередачи и т. п. В этом случае на веб-странице размещён электронный плеер с инструментами для начала или приостановки прослушивания аудиозаписи и регулирования громкости (рис. 11.3).

Рис. 11.3

Интересно

Термин **подкаст** состоит из названия портативного проигрывателя музыки *iPod* и слова *broadcast* — от англ. трансляция, радиовещание.

ДЕЙСТВУЕМ

Упражнение 1. Прослушивание аудиозаписей онлайн.
Задание. Прослушайте фрагмент аудиозаписи поэмы Леси Украинки «Давня казка».

1. Откройте окно браузера и в строке адреса введите *https://lucaster.podfm.ru/ukraudio/140/*
2. Прослушайте с помощью электронного плеера (рис. 11.3) фрагмент аудиокниги «Давня казка» Леси Украинки в исполнении Ады Роговцевой.
3. Определите, как изменить громкость воспроизведения, приостановить прослушивание.
4. Определите длительность аудиозаписи.

5. Как искать и просматривать видео на *YouTube*?

Для начала работы на *YouTube* нужно загрузить браузер и в строке адреса ввести *www.youtube.com*. Главная страница сайта, как и поисковых служб, содержит строку ввода ключевых слов. При её использовании поиск происходит по указанным ключевым словам среди размещённых на *YouTube* видеозаписей (рис. 11.4).

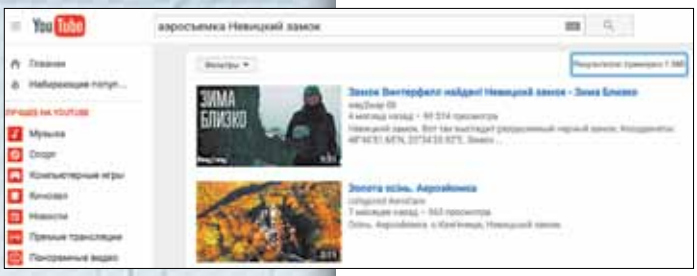
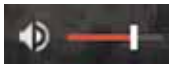


Рис. 11.4

Если вы нашли интересное видео, то с его помощью можно найти и другие: в правой части окна находятся ссылки на другие видео, размещённые этим пользователем, а также похожие видео.

В нижней части области воспроизведения видеозаписи (рис. 11.5) находятся элементы управления воспроизведением и просмотром: кнопки для начала  или приостановки воспроизведения  , регулирования громкости  , перехода к режиму домашнего кинотеатра  или полноэкранному режиму  . Перейти к просмотру следующего видео выбранной тематики можно с помощью элемента  .

Инструмент *Настройка*  выбирают для изменения качества видео в зависимости от скорости интернет-соединения (рис. 11.6).

Рис. 11.5

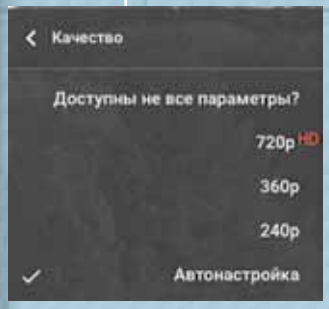


Рис. 11.6

Видеозаписи, размещённые на *YouTube*, содержат потоковое видео, поэтому их можно просматривать в режиме реального времени. Если скорость загрузки данных из Интернета недостаточна для комфортного просмотра, можно его приостановить с помощью кнопки *Пауза* и дождаться загрузки хотя бы третьей части или половины видео.

Упражнение 2. Поиск и просмотр видео на *YouTube*.
Задание. Найдите и просмотрите на *YouTube* видео «Золотая осень. Аэросъёмка».

1. Откройте окно браузера и в адресной строке введите *www.youtube.com*.
2. На главной странице в строку ввода введите ключевые слова *аэросъёмка Невицкий замок*.
3. Выберите один из найденных результатов поиска (рис. 11.4).
4. Просмотрите видеозапись. В случае необходимости приостановите воспроизведение и дождитесь частичной загрузки видео.
5. Среди похожих видео найдите другие видеозаписи о замках Карпат или аэросъёмке. Просмотрите найденные записи.

6. Какие программы используются для обработки аудиозаписей?

Создавать звуковые файлы и вносить изменения в аудиозаписи можно с помощью **звуковых редакторов** или **аудиоредакторов**. Такие программы используют для записи звукового файла с микрофона, объединения нескольких аудиозаписей, вырезания и сохранения фрагмента аудиозаписи в отдельном файле, удаления паузы, избавления от шума и т. п.

Существуют как бесплатные, так и более мощные платные звуковые редакторы. Примеры звуковых редакторов: *Audacity* (рис. 11.7), *mp3DirectCut*, *Ocenaudio*, *Adobe Audition, Acoustica, Audio Edit Deluxe* и т. п.

Звуковые данные в среде аудиоредактора графически представляются в волновой форме. Область с таким графическим изображением в звуковом редакторе называется **треком**, или **звуковой дорожкой**.

ДЕЙСТВУЕМ

Упражнение 3. Ознакомление с назначением и основными приёмами работы с аудиоредактором *Audacity*.

Задание. Определите назначение некоторых элементов управления аудиоредактора *Audacity*.

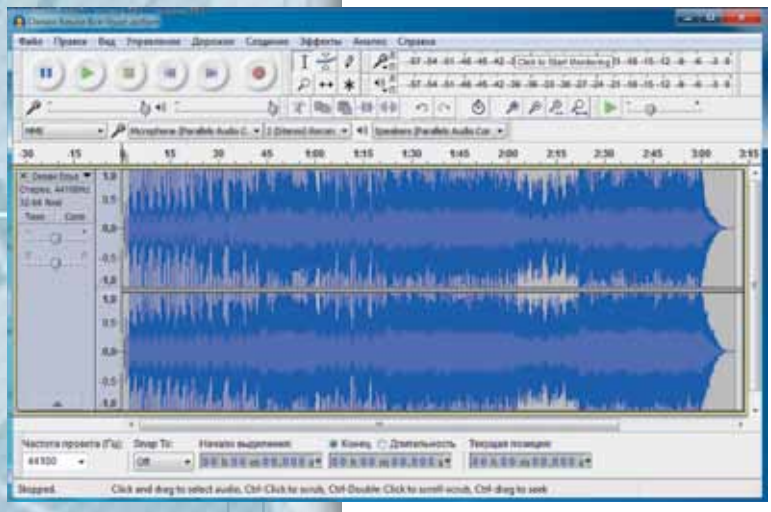


Рис. 11.7

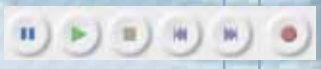


Рис. 11.8

1. Выполните команду *Пуск/Все программы/Audacity*.
2. Рассмотрите элементы окна программы.
3. В меню *Файл* выберите команду *Открыть* и в папке *Объекты мультимедиа\Аудио* откройте файл *Океан Ельзи Все буде добре.mp3*.
4. Определите назначение инструментов (рис. 11.8).
5. Прослушайте аудиозапись, воспользовавшись инструментом *Воспроизведение*. Определите, когда начинается последний припев.
6. Выполняя протягивание мышью, выделите фрагмент аудиозаписи с последним припевом.
7. В нижней части окна просмотрите значение времени для свойств *Начало выделения* и *Конец*.
8. Ознакомьтесь с командами, содержащими разные пункты меню программы.

7. С помощью каких программ можно создать собственный видеоклип?

Для создания видеоклипов используются видеоредакторы.

Видеоредактор — это программа, содержащая набор инструментов, с помощью которых создают и редактируют видеофайлы на компьютере.

Возможности и наборы инструментов для обработки видеофайлов у разных видеоредакторов отличаются.

В состав операционной системы *Windows 7* входит видеоредактор *Киностудия* (или *Windows Live Movie Maker*). Используя его инструменты, можно удалять лишние кадры из видеозаписи, располагать видеофрагменты в любой последовательности, добавлять музыкальные файлы, голосовое сопровождение, титры и т. п. Видеоредактор *Киностудия* был создан на смену программе *Windows Movie Maker,* входящей в состав операционной системы *Windows XP*.

Существуют и другие видеоредакторы, в частности *VirtualDub, Pinnacle Studio*, *Free Video Editor*, *MS Producer*, *Adobe After Effects*, *Adobe Premiere*, *Ulead MediaStudio*, *SONY Vegas Pro* и т. п. С помощью программы *Camtasia Studio* можно также записывать видео с экрана компьютера.

С помощью видеоредактора создаётся **проект** — файл, содержащий сведения о порядке расположения и времени воспроизведения аудио- и видеоклипов, видеопереходах, видеоэффектах, названиях, титрах и т. п. После сохране-

ния проекта его файл можно открыть позже в среде видеоредактора и внести изменения. Проекты, созданные в программе *Киностудия*, имеют расширение *wlmp*, а в программе *Windows Movie Maker* — *mswmm*.

Готовый проект может быть сохранён как **фильм** — видеофайл в соответствующем формате, например WMV либо MPEG4. Фильм можно сохранить на компьютере или компакт-диске, отправить по электронной почте либо разместить в Интернете.

8. Какие особые элементы интерфейса содержит программа *Киностудия*?

Кроме стандартных элементов окна, в частности, строки заголовка и ленты, где находятся вкладки с инструментами, в окне видеоредактора *Киностудия* есть также особые элементы, присущие видеоредакторам. **Область предварительного просмотра** используют для просмотра как отдельных клипов, так и всего проекта перед сохранением, а область, в которой создаются и монтируются проекты, используют для просмотра и изменения последовательности клипов проекта (рис. 11.9). Аудиоклипы, добавленные к проекту, отображаются ниже добавленных видеоклипов.

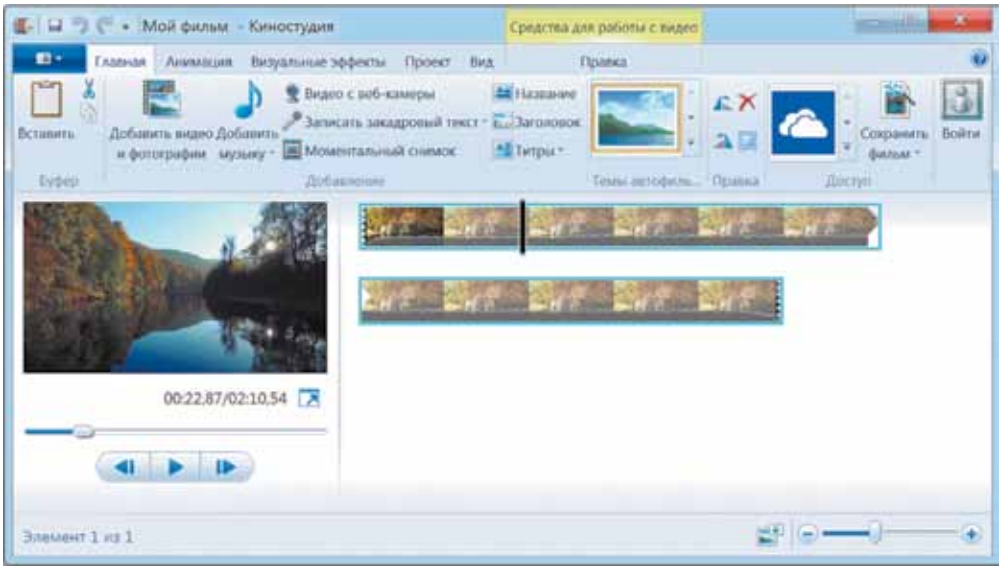


Рис. 11.9

ДЕЙСТВУЕМ

Упражнение 4. Назначение и основные приёмы работы с видеоредактором *Киностудия*.

Задание. Просмотрите видеозапись о назначении и основных приёмах работы с видеоредактором *Киностудия*.

1. Откройте папку *Объекты мультимедиа\Видео* и запустите на воспроизведение файл *Обзор программы Киностудия.mp4*.
2. Просмотрите видеозапись.
3. Объясните, какие действия можно выполнить с помощью видеоредактора *Киностудия*.

Интересно

Если предусматривается обработка видео, то компьютер должен соответствовать таким минимальным системным требованиям: процессор мощностью не ниже 600 Мгц, например, *Intel Pentium III, AMD Athlon* и т. п., 128 Мб ОЗУ, 2 Гб свободного пространства на диске, наличие устройства звукозаписи (для записи звука из внешних источников) и устройства видеозаписи (для записи видео из внешних источников). Основным требованием для возможности переноса видео с видеокамеры является наличие разъёма *FireWire*, карты видеозахвата или видеовхода на видеокарте.

Упражнение 5. Элементы интерфейса видеоредактора *Киностудия*.
Задание. Определите назначение некоторых элементов управления видеоредактора *Киностудия*.

1. Выполните команду *Пуск/Все программы/Киностудия*.
2. Рассмотрите элементы окна программы.
3. С помощью задержки мыши определите назначение инструментов на разных вкладках ленты.
4. С помощью задержки мыши определите назначение кнопок, расположенных в нижней части области предварительного просмотра.
5. Откройте меню *Файл* с помощью кнопки  в левом верхнем углу окна и ознакомьтесь с командами работы с файлами в программе *Киностудия*. Какие команды являются общими и различными в программе *Киностудия* и текстовом процессоре *Microsoft Word*?

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 11* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Используя Википедию, найдите сведения об особенностях:
   а) аудиоформатов;
   б) видеоформатов.
   Сравните сведения, приведённые на русском, украинском и английском языках (при необходимости воспользуйтесь переводчиком). Обсудите в парах.
2. Обсудите преимущества и недостатки прослушивания аудиозаписей онлайн.
3. Найдите в Интернете веб-страницы любимых радиостанций. Обсудите, какие из них позволяют прослушивать радиопередачи онлайн.
4. Можно ли импортировать в проект, создаваемый с помощью видеоредактора *Киностудия*, видеозапись в формате FLV? Обсудите в парах.
5. Найдите и установите на компьютер одну из бесплатных программ-конвертеров видео. Ознакомьтесь с её интерфейсом и возможностями. Выясните, из каких форматов и в какие можно преобразовывать видео с помощью этой программы. Научите друг друга пользоваться программой.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Используя словарь или переводчик, переведите полные названия аудио- и видеоформатов на русский язык. При необходимости обратитесь к свободной энциклопедии Википедии.
2. Откройте веб-страницу *http://www.elecard.com/mpeg/faq/index.php* (или файл *Экспертная группа MPEG* в папке *Объекты мультимедиа*). Ознакомьтесь с предложенными материалами и дайте ответы на вопросы:
   а) как часто собирается *экспертная группа по движущемуся изображению* (MPEG);
   б) во сколько раз происходит сжатие видео при использовании формата MPEG;
   в) как работает MPEG видео?

3. Ознакомьтесь с особенностями создания и размещения в Интернете подкаста. Воспользуйтесь ссылкой *https://podfm.ru/topodcasters/*.
4. Найдите на *YouTube* видеозаписи, отображающие процесс работы или результаты деятельности людей разных профессий: ландшафтного дизайнера, скульптора, телеведущего, журналиста, кулинара и т. п. Сохраните ссылки на найденные страницы в *Избранное*.

ИССЛЕДУЕМ

1. Определите, как можно сохранить в файле видео, найденное в *YouTube*. По результатам исследования подготовьте презентацию.
2. Определите, файлы каких типов можно использовать для создания видеоклипа с помощью программы *Киностудия*. Воспользуйтесь материалами по ссылке *http://windows.microsoft.com/uk-ua/windows-live/movie-maker-file-types-faq*.

ПОЛЕЗНЫЕ ССЫЛКИ

Инструкции по работе с программой *Киностудия*: *https://support.microsoft.com/ru-ru/help/18614/windows-essentials#*

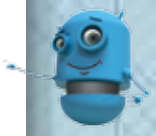
Путеводитель по звуковым редакторам: *http://www.ixbt.com/soft/wave-editors-part1.shtml*

12. СОЗДАНИЕ И НАСТРОЙКА ВИДЕО И АУДИО

ВЫ УЗНАЕТЕ:

- как добавить в проект мультимедийные объекты и текстовые надписи;
- как разделить видеоклип на два и скрыть начало или конец видеоклипа;
- как добавить видеопереходы между клипами или изображениями и визуальные эффекты;
- как добавить к видеоклипу музыкальное сопровождение и настроить временные параметры видео и аудио;
- как сохранить созданный видеоклип в формате видео;
- как опубликовать видео на *YouTube*;
- как можно вносить изменения в видео с помощью онлайн-сервисов.

ИЗУЧАЕМ



1. Как добавить в проект мультимедийные объекты и текстовые надписи?

Для создания собственного видео можно использовать готовые видеоклипы, содержащиеся в файлах, или захватить видео непосредственно с видеокамеры или веб-камеры. Видеоклип, создаваемый с помощью программы *Киностудия*, может содержать видеофрагменты, аудиозаписи и статические изображения. Для добавления в проект мультимедийных объектов из

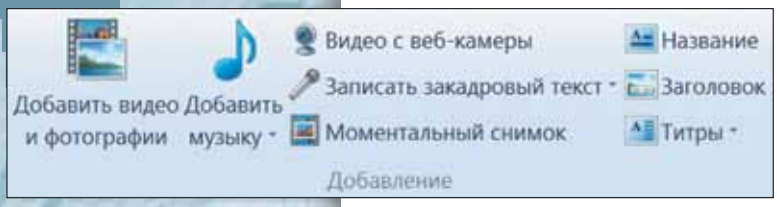


Рис. 12.1

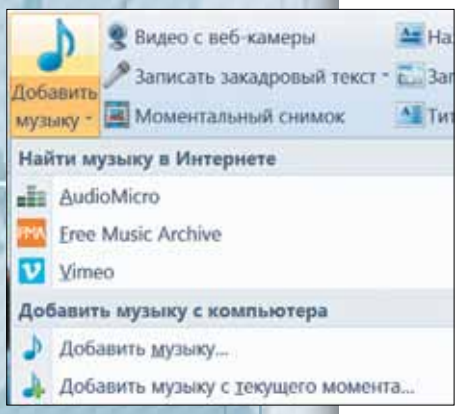


Рис. 12.2

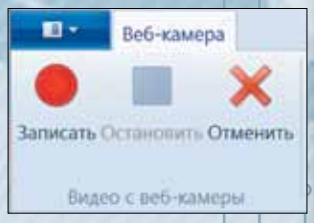


Рис. 12.3

файлов нужно воспользоваться инструментами из группы *Добавление* вкладки *Главная* (рис. 12.1).

Если мультимедийные объекты сохранены в форматах, которые программа *Киностудия* не поддерживает, для использования в проекте их следует сначала конвертировать в один из поддерживаемых форматов.

Добавлять музыкальное сопровождение к видеоклипу можно как из аудиофайлов на компьютере, так и из Интернета (рис. 12.2).

Видеоклип можно дополнить звуковым сопровождением в виде дикторского текста, записанного с использованием микрофона. Для записи и добавления в проект видео или дикторского текста необходимо выбрать соответствующий инструмент в группе *Добавление* на вкладке *Главная* и воспользоваться инструментом *Записать* для начала записи (рис. 12.3).

В процессе записи дикторского текста в области предварительного просмотра воспроизводится видеозапись, позволяющая синхронизировать комментарий с видео. Для завершения записи следует нажать кнопку *Остановить*, при этом будет предложено ввести имя файла, в который будет сохранён комментарий. Созданная видео- или аудиозапись хранится в файле и добавляется к проекту.

Совокупность добавленных мультимедийных объектов становится содержимым проекта и будущим фильмом.

Используя инструменты *Название*, *Заголовок*, *Титры* из группы *Добавление* вкладки *Главная* (рис. 12.1), можно добавить в проект текстовые надписи: название фильма, пояснения или комментарии к отдельным клипам, титры в конце фильма со сведениями об авторах, использованных материалах и т. п.

ДЕЙСТВУЕМ

Упражнение 1. Создание видеоклипа, состоящего из нескольких видеозаписей.

Задание. Создайте проект о музыкальных фонтанах с видеоклипами фонтанов в Одессе, Барселоне, Праге, Лас-Вегасе.

1. В своей структуре папок создайте папку *Мультимедиа*.
2. Откройте окно видеоредактора *Киностудия* и выберите инструмент *Добавить видеозаписи и фотографии*.
3. Добавьте в проект все видеозаписи из папки *Объекты мультимедиа\Видео\Музыкальные фонтаны*.
4. Сохраните созданный проект с именем *Музыкальные фонтаны* в папке *Мультимедиа* своей структуры папок. Для этого нажмите кнопку  и выполните команду *Сохранить проект*.

Упражнение 2. Добавление названия фильма и названий клипов в проект.

Задание. В проект *Музыкальные фонтаны*, находящийся в папке *Мультимедиа* своей структуры папок, добавьте названия фильма и некоторых клипов.

1. В папке *Мультимедиа* своей структуры папок откройте проект *Музыкальные фонтаны*.
2. На вкладке *Главная* в группе *Добавить* выберите инструмент *Название*.
3. В области предварительного просмотра в текстовой надписи введите фразу *Музыкальные фонтаны*. На вкладке *Форматирование*, появляющейся при обработке текстовых надписей, измените параметры форматирования текста по своему усмотрению: задайте цвет символов, их размер, начертание и т. п.
4. Выделите видеоклип *Фонтан Одесса* и на вкладке *Главная* в области *Добавить* выберите инструмент *Заголовок*.
5. Введите название *Одесса* и установите параметры форматирования для текстовой надписи.
6. Аналогично добавьте названия городов в начале каждой следующей видеозаписи.
7. Просмотрите созданный проект.
8. Выполните команду *Сохранить проект*.

2. Как разделить видеоклип на два и скрыть начало или конец видеоклипа?

Добавленные в проект видеоклипы иногда нужно дополнительно отредактировать — разделить на несколько частей и скрыть начало или конец клипа.

Если время воспроизведения клипа достаточно продолжительное и при этом он требует редактирования — удаления кадров или добавления видеоперехода внутри клипа, такой клип можно разделить. Для разделения клипа на два нужно переместить индикатор воспроизведения в место клипа, где его нужно разделить, и на вкладке *Правка* выбрать инструмент *Разделить* (рис. 12.4).

Клипы могут содержать фрагменты, которые не нужно воспроизводить в фильме. Если такие фрагменты расположены в начале или в конце клипа, их можно скрыть — «обрезать» начало либо конец клипа. Для таких действий используют инструмент *Средство усечения* на вкладке *Правка*. У каждого выделенного клипа есть начальная и конечная точки (рис. 12.5).

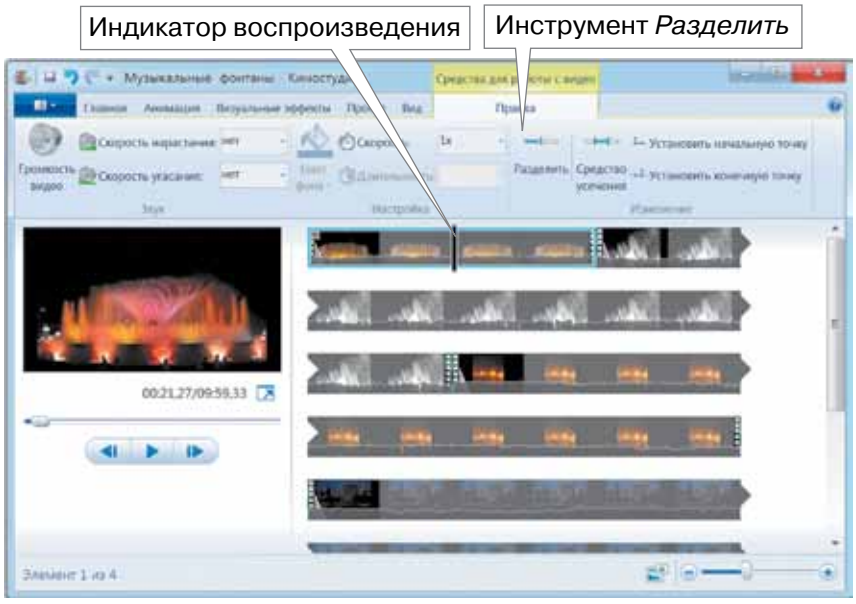


Рис. 12.4

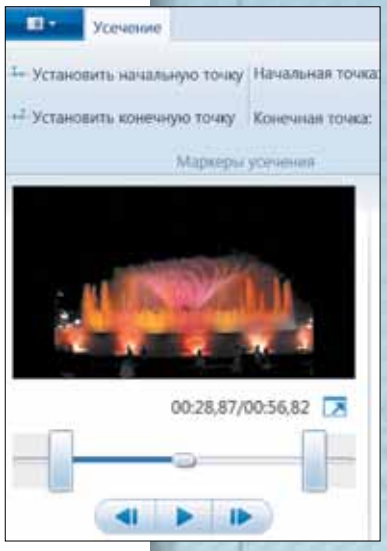


Рис. 12.5

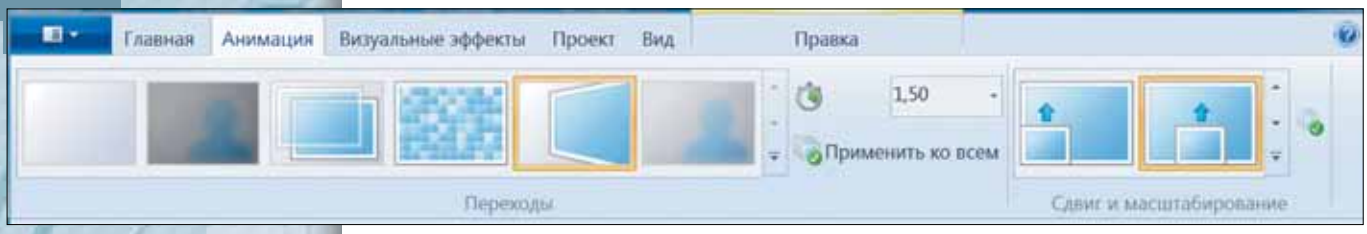


Рис. 12.6

3. Как добавить видеопереходы между клипами или изображениями и визуальные эффекты?

По аналогии с анимационными эффектами изменения слайдов в компьютерных презентациях при создании фильма можно добавлять видеопереходы между отдельными клипами. Перечень видеопереходов находится на вкладке *Анимация* (рис. 12.6). Для добавления видеоперехода в проект достаточно выделить видеоклип и выбрать один из эффектов перехода или сдвига и масштабирования.

Иногда реализовать замысел автора помогают визуальные эффекты, которые можно добавлять к клипам. Например, с помощью эффекта «старого фильма», будто записанного на плёнке, можно воспроизвести клип в чёрно-белом виде и т. п. Перечень всех доступных эффектов находится на вкладке *Визуальные эффекты*. Прежде чем применить определённый видеоэффект к клипу, просмотрите, как он будет влиять на воспроизведение. Для этого достаточно выделить видеоэффект и увидеть его действие в области предварительного просмотра.

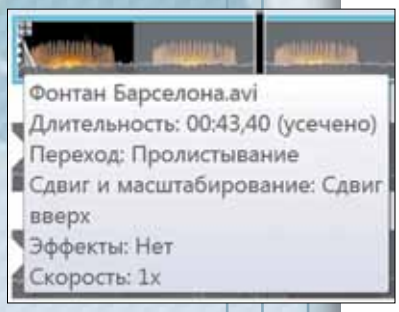


Рис. 12.7

Для добавления к клипу видеоэффекта следует выделить видеоклип и выбрать соответствующий эффект. Узнать, к каким клипам добавлены видеоэффекты или видеопереходы, можно с помощью всплывающей подсказки — если навести указатель мыши на клип и выполнить задержку, отобразится список текущих параметров и их значений (рис. 12.7).

4. Как добавить к видеоклипу музыкальное сопровождение и настроить временные параметры видео и аудио?

Видеозаписи, записанные на видеокамеру или импортированные из файлов, могут содержать и звук. Если видеозапись не содержит звука или имеющийся звук воспроизводить не нужно, можно добавить к видеоклипу другое музыкальное сопровождение или звуковой комментарий.

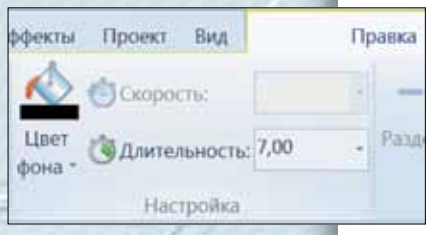


Рис. 12.8

Очень редко возникают ситуации, когда длительность видео и аудио совпадает, поэтому возникает необходимость настраивать временные параметры. Получить видеозапись или аудиоклип нужной длительности можно разными способами: скрыть начало или конец определённых видео- или аудиоклипов, изменить скорость воспроизведения видео в группе *Настройка* на вкладке *Правка* (рис. 12.8) соответственно для уменьшения или увеличения длительности клипа, добавить к видеоклипу заголовок, титры или дополнительные изображения и т. п. Длительность отображения изображений, заголовка и титров можно изменять с помощью соответствующего параметра (рис. 12.8) в группе *Настройка*.

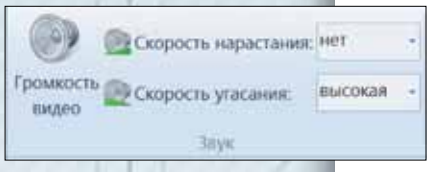


Рис. 12.9

Если длительность аудиозаписи больше, чем видеоряда, аудиозапись будет автоматически «обрезана» в конце видеоклипа. Если аудиозапись будет воспроизводиться не полностью, к ней можно применить эффекты *Скорость нарастания* или *Скорость угасания* в группе *Звук* на вкладке *Правка* (рис. 12.9). Кроме того, разрешается регулировать громкость воспроизведения звука или выключить звук для видеоклипа.

ДЕЙСТВУЕМ

Упражнение 3. Добавление в видеоклип музыкального сопровождения и настройка временных параметров видео и аудио.

Задание. В проект *Утро*, находящийся в папке *Мультимедиа* своей структуры папок, импортируйте музыкальный файл *Григ Утро.mp3.* Добавьте аудиоклип и настройте временные параметры видео и аудио.

1. Откройте окно видеоредактора *Киностудия* и выберите инструмент *Добавить видеозаписи и фотографии*.
2. Добавьте в проект видеозапись *Утро.avi* из папки *Объекты мультимедиа\Видео*.
3. Выберите инструмент *Добавить музыку* и импортируйте звукозапись *Григ Утро.mp3* из папки *Объекты мультимедиа\Аудио.*
4. Чтобы увеличить длительность видеоклипа *Утро* в 4 раза (замедлить его), выделите этот клип и на вкладке *Правка* в группе *Настройка* установите *Скорость* — 0,25.
5. Выделите аудиоклип *Григ Утро* и на вкладке *Правка* в группе *Аудио* установите *Скорость угасания* — высокая (рис. 12.9).
6. Просмотрите созданный видеоклип.
7. Сохраните созданный проект в папке *Мультимедиа* своей структуры папок.

5. Как сохранить созданный видеоклип в формате видео?

По завершению работы над проектом можно сохранить готовый видеоклип как фильм в формате видео. Для этого следует выполнить команду *Файл/Сохранить фильм* (рис. 12.10), которая запускает *Мастер сохранения фильмов*. В зависимости от того, где нужно сохранить фильм, может быть выбран один из предложенных вариантов. На следующих шагах работы *Мастера сохранения фильмов* предлагается указать имя файла и параметры, влияющие на качество видео и объём файла, но можно не изменять параметры, заданные по умолчанию.

Сохранённый фильм можно опубликовать в Интернете (рис.12.11).

Рис. 12.11

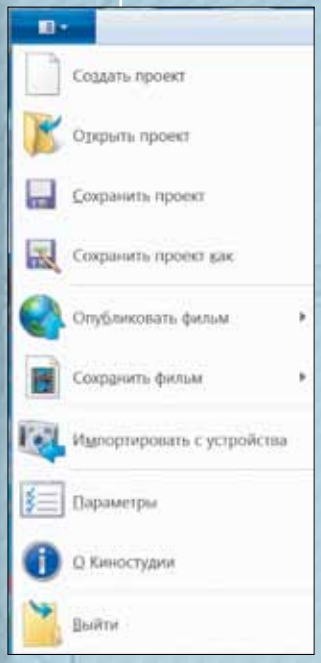


Рис. 12.10

ДЕЙСТВУЕМ

Упражнение 4. Сохранение созданного проекта видеоклипа в формате видео.

Задание. Сохраните проект *Музыкальные фонтаны* как фильм.

1. В папке *Мультимедиа* своей структуры папок откройте проект *Музыкальные фонтаны*.
2. В меню *Файл* выберите команду *Сохранить фильм* и параметры, рекомендованные для этого проекта.
3. В окне сохранения файла выберите папку *Мультимедиа* своей структуры папок, укажите название фильма *Музыкальные фонтаны*, выберите расширение файла *wmv* и нажмите кнопку *Сохранить*.

Рис. 12.12

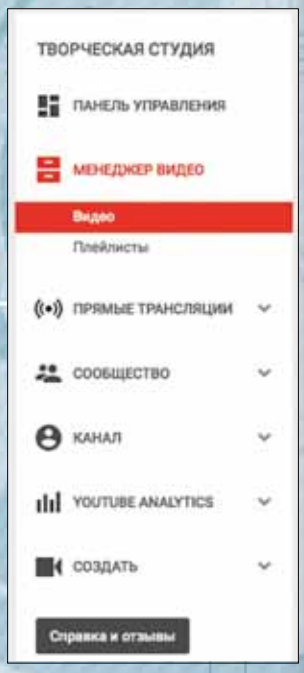


Рис. 12.13

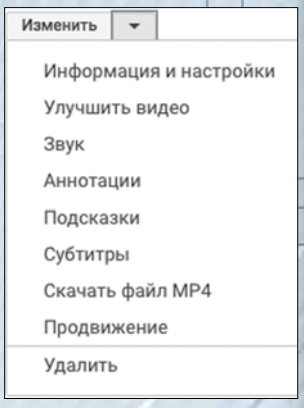


Рис. 12.14

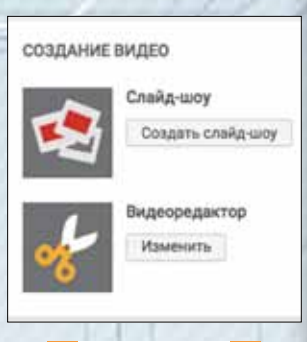


Рис. 12.15

6. Как опубликовать видео на *YouTube*?

Искать и просматривать видео на *YouTube* можно и без регистрации. Однако для загрузки видео необходимо воспользоваться собственной учётной записью *Google*. Чтобы получить возможность загрузки видео, следует войти в свою учётную запись и выбрать команду *Добавить видео* (рис. 12.12).

Перед загрузкой видеозаписи нужно создать канал пользователя. Это можно сделать в *Творческой студии* или при первом использовании команды *Добавить видео*: будет предложено создать канал пользователя, где будут храниться загруженные видеозаписи.

Видео, содержащееся в файле, можно выбрать в структуре папок или просто перетянуть мышью в указанную область.

Прежде чем опубликовать загруженное видео, необходимо указать его название, описание, ключевые слова и другие настройки для видео, по которым другие пользователи смогут его найти.

Просмотреть список загруженных видео на канале и внести в них изменения можно с помощью *Менеджера видео* в *Творческой студии* (рис. 12.13).

Каждое загруженное видео можно редактировать или удалить (рис. 12.14).

7. Как можно вносить изменения в видео с помощью онлайн-сервисов?

Кроме видеоредакторов в виде программ, устанавливаемых на компьютер, существуют также онлайновые видеоредакторы, например, *FileLab Video Editor* (*https://www.filelab.com/ru/video-editor*) и собственный видеоредактор на *YouTube* — *https://www.youtube.com/editor*. Перейти к видеоредактору *YouTube* можно также со страницы для загрузки видео (рис. 12.15).

Как и в среде видеоредактора *Киностудия*, в онлайновом видеоредакторе *YouTube* разрешается добавлять видеофрагменты, изображения, аудиозаписи, видеопереходы и текстовые надписи (рис. 12.16).

Видеофрагменты можно разделить на два, обрезать с начала или с конца, выполнить быстрые настройки, повернуть, если видеозапись снимали вертикально и т. п.

Рис. 12.16

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 12* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Рассмотрите разные способы усечения видеозаписи: перемещение ползунков в области просмотра, задание времени начальной и конечной точки, инструменты *Установить начальную точку* и *Установить конечную точку* (рис. 12.5). Обсудите, как использовать каждый из этих способов. Какой способ вы считаете более удобным?
2. Что общего и различного имеют область предварительного просмотра видеоредактора *Киностудия* и окно программы *Проигрыватель Windows Media*? По результатам сравнения постройте диаграмму Венна. Обсудите в парах.
3. Обсудите преимущества и недостатки редактирования видеоклипа с помощью онлайнового видеоредактора по сравнению с видеоредактором, установленным на компьютере.
4. Ознакомьтесь с дополнительными настройками, которые можно изменять при загрузке видео на *YouTube* (рис. 12.17). Обсудите в парах, в каких случаях целесообразно изменять такие настройки.

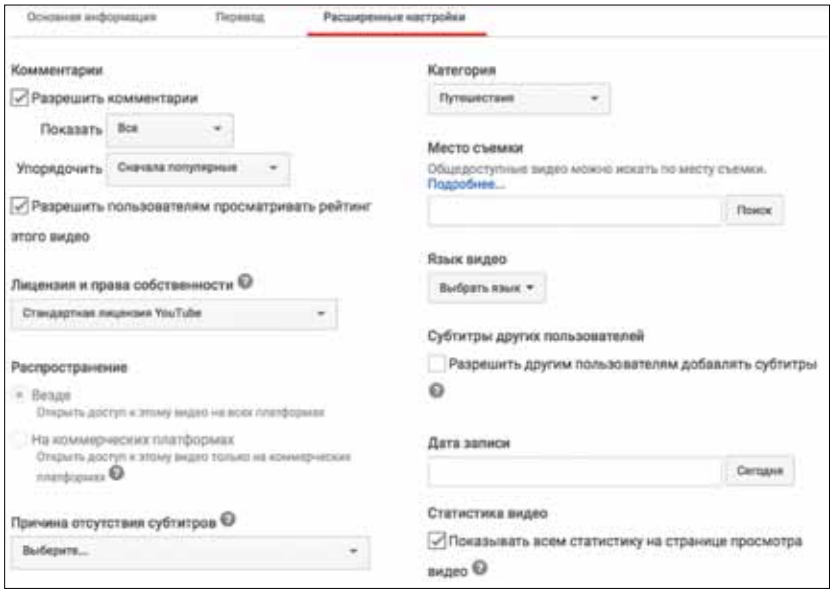


Рис. 12.17

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Определите, в каких папках по умолчанию будет храниться видео, захваченное с видеокамеры, и аудиозапись при записи закадрового текста.
2. Определите, на каких сервисах можно разместить в Интернете видео, созданное в программе *Киностудия* (рис. 12.11).

3. Откройте проект *Музыкальные фонтаны*, находящийся в папке *Мультимедиа* своей структуры папок. Добавьте видеопереходы между клипами на своё усмотрение.
4. Создайте проект *Прогулка по Киеву*, в который импортируйте видеозапись *Прогулка по Киеву.wmv*, и примените видеоэффекты для отображения видео в оттенках серого. Определите, распространяется ли на весь клип действие видеоэффекта, добавленного в клип, либо только на его начало или определённую часть.
5. Создайте видеоэкскурсию по своему городу или селу. Используйте фотографии, видеозаписи прогулки, музыкальное сопровождение. Примените видеопереходы и видеоэффекты и создайте название клипа и титры. Сохраните фильм в файле с именем *Экскурсия.wmv* в папке *Мультимедиа* своей структуры папок. В качестве примера рассмотрите видеозапись *Прогулка по Киеву.wmv*, находящуюся в папке *Объекты мультимедиа\Видео*.
6. Создайте проект *Моё увлечение*, для которого подготовьте фотографии и видеозаписи. Используйте видеопереходы между изображениями и видеозаписями, добавьте название фильма и титры. Готовый проект сохраните как файл фильма в папке *Мультимедиа* своей структуры папок.

ИССЛЕДУЕМ

Определите, какие действия можно выполнить с клипом при помощи команд контекстного меню (рис. 12.18).

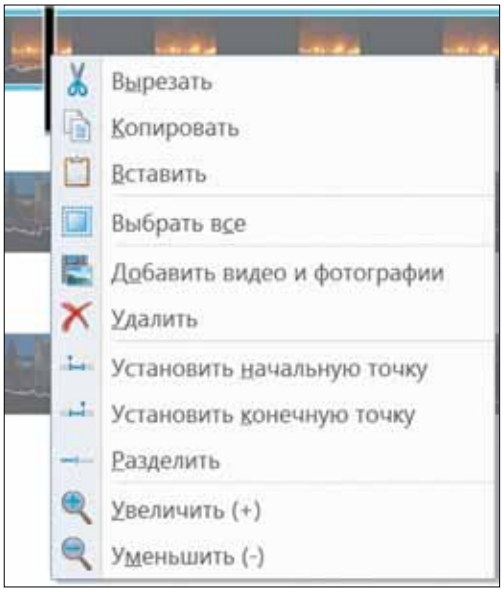


Рис. 12.18

ПОЛЕЗНЫЕ ССЫЛКИ

FileLab Video Editor:
https://www.filelab.com/ru/video-editor
Собственный видеоредактор на *YouTube*:
https://www.youtube.com/editor

13. ПРАКТИЧЕСКАЯ РАБОТА 6

СОЗДАНИЕ ВИДЕОКЛИПА. ДОБАВЛЕНИЕ ВИДЕОЭФФЕКТОВ, НАСТРОЙКА ВРЕМЕННЫХ ПАРАМЕТРОВ АУДИО- И ВИДЕОРЯДА

ВСПОМНИТЕ

- Как добавлять к видеоклипу аудио- и видеофрагменты из внешних источников;
- как синхронизировать видеоряд с аудиорядом;
- как добавлять видеоэффекты к видеоклипу и настраивать переходы между его фрагментами.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 6.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Семь чудес Древнего мира (*9 баллов*)

Найдите в Интернете сведения о Семи чудесах Древнего мира. Используя видеозаписи в папке *Объекты мультимедиа\Семь чудес света*, создайте видеоклип о Семи чудесах света. Добавьте названия фильма и каждого из клипов. Добавьте видеопереходы между клипами и видеоэффекты.

Задание 2. Ансамбль Вирского (*10 баллов*)

Создайте новый проект в программе *Киностудия*. Добавьте в проект видеозаписи, находящиеся в папке *Объекты мультимедиа\Ансамбль Вирского*. Создайте видеоклип «*Украинские танцы в исполнении ансамбля Вирского*». Для этого из каждого танца выберите наиболее яркие логично завершённые фрагменты (используйте разделение клипа либо усечение начала или/и конца клипа). Добавьте к каждому фрагменту название танца и видеопереходы между фрагментами.

Задание 3. Ландшафтный дизайн (*12 баллов*)

В папке *Объекты мультимедиа\Ландшафтный дизайн* просмотрите видеозаписи *Ландшафтный дизайн.avi* и *Ландшафтный дизайн Цветы.wmv*. Найдите в Интернете интересные изображения, относящиеся к ландшафтному дизайну. Подберите музыкальную композицию в формате, поддерживаемом видеоредактором *Киностудия*, для звукового сопровождения видеоклипа. Создайте проект о ландшафтном дизайне, в который добавьте найденные изображения и отдельные видеофрагменты из предложенных видеозаписей. Примените видеопереходы и видеоэффекты. Добавьте аудиозапись для звукового сопровождения видеоклипа.

Задание 4. История развития компьютеров (*12 баллов*)

Используя видеозаписи в папке *Объекты мультимедиа\История развития компьютеров*, создайте видеозапись об электронной вычислительной машине ENIAC. Для этого создайте новый проект в программе *Киностудия*, добавьте к проекту видеозапись *История развития компьютеров.mp4* и установите начальную и конечную точки так, чтобы воспроизводился только фрагмент об электронной вычислительной машине ENIAC. Добавьте к проекту видеозапись *Интересные факты о первом компьютере.mp4* и закадровый текст, где будут озвучены интересные факты, отображённые в видеозаписи.

14. ПРАКТИЧЕСКАЯ РАБОТА 7

РАЗМЕЩЕНИЕ АУДИО- И ВИДЕОМАТЕРИАЛОВ В ИНТЕРНЕТЕ

ВСПОМНИТЕ

- Как размещать аудио- и видеоматериалы в Интернете.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 7*.

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Создание собственного канала на *YouTube* (*6 баллов*)

Откройте окно браузера и в строке адреса введите *https://www.youtube.com*. Войдите в свою учётную запись *Google* с помощью кнопки *Войти* (рис. 12.14). Если у вас ещё нет такой учётной записи, создайте её, воспользовавшись ссылкой *Создать учётную запись*. Выберите объект  в правом верхнем углу окна и нажмите кнопку *Творческая студия* (рис. 14.1). Выберите ссылку *Создать канал*. В появившемся окне введите свою фамилию и имя, ознакомьтесь с условиями использования *YouTube* и нажмите кнопку *Создать канал* (рис. 14.2).

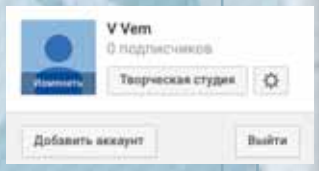


Рис. 14.1

Задание 2. Публикация видео на *YouTube* (*11 баллов*)

На странице сервиса *YouTube* убедитесь, что вы вошли в свою учётную запись, и нажмите кнопку *Добавить видео* в правом верхнем углу окна. Сверните окно браузера и откройте папку *Мультимедиа* в своей структуре папок. Разверните окно браузера и измените размещение окон так, чтобы окно браузера и окно папки находились на экране рядом. Перетяните мышью файл *Музыкальные фонтаны* в область для загрузки файлов (рис. 14.3). Введите название, описание видео и ключевые слова для поиска, при необходимости измените другие настройки. Нажмите кнопку *Опубликовать*. Определите *URL*-адрес страницы, на которой опубликовано ваше видео. Отправьте учителю по электронной почте адрес опубликованного видео.

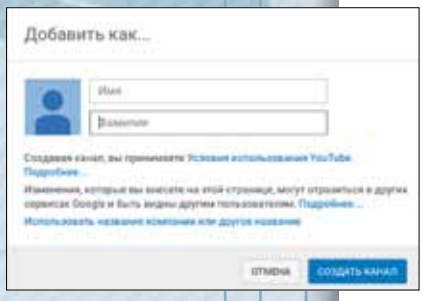


Рис. 14.2

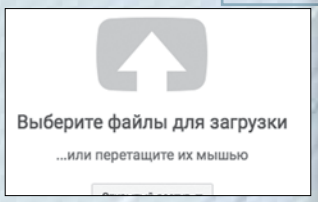


Рис. 14.3

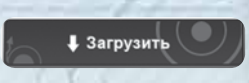
Задание 3. Создание аудиозаписи (*6 баллов*)

С помощью аудиоредактора запишите с микрофона отрывок поэтического или прозаического произведения современного украинского писателя. При необходимости удалите паузы из аудиозаписи. Сохраните созданную аудиозапись в файле с именем *Отрывок произведения.mp3* в папке *Практическая работа 7.*

Задание 4. Публикация аудиозаписи (*4 балла*)

Зарегистрируйте собственную учётную запись на портале *http://meta.ua*. Опубликуйте созданную аудиозапись на сервисе размещения аудио *http://audio.meta.ua*.

Для этого воспользуйтесь инструментом *Загрузить*.



В открывшемся окне при необходимости измените настройки, указанные в шагах 1 и 2, и на шаге 3 выберите файл *Отрывок произведения.mp3* из папки *Практическая работа 7*. Дождитесь, пока файл будет полностью загружен. Это можно определить с помощью соответствующей шкалы.

. Для публикации файла нажмите кнопку *Загрузить файлы*.

РАЗДЕЛ 5

ОСНОВЫ СОБЫТИЙНО- И ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

15. ЯЗЫК И СРЕДА ПРОГРАММИРОВАНИЯ

ВСПОМНИТЕ:

- что называется алгоритмом и программой;
- что или кто может быть исполнителем алгоритма;
- в какой форме можно представить алгоритм;
- что называется средой выполнения алгоритма;
- как создают программы и проекты в среде *Скретч*.

ВЫ УЗНАЕТЕ:

- что такое язык программирования;
- чем отличаются языки программирования;
- какие средства необходимы для выполнения созданных программ;
- какие среды программирования используются для создания программ;
- как работать с проектом в среде программирования *PyCharm*.

ИЗУЧАЕМ

1. Что такое язык программирования?

Вы уже знаете, что алгоритмы для исполнителей описывают различными способами и используются раздичные формы их представления. Если алгоритм создаётся для человека, который будет его выполнять, то, как правило, алгоритм представляют в словесной или графической форме. Для описания алгоритмов, ориентированных для исполнения компьютером, используют **язык программирования**. Алгоритм, записанный с помощью языка программирования, называется **программой**.

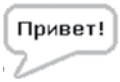
Язык программирования — это система обозначений для точного описания алгоритма, который нужно выполнить с помощью компьютера.

Язык программирования, как и любой другой язык, имеет следующие составляющие:

- набор символов, из которых создают слова и предложения, — **алфавит**;
- совокупность специальных слов, имеющих однозначное объяснение и применение, — **словарь**;
- систему правил составления базовых конструкций языка — **синтаксис**;
- правила **семантики**, объясняющие, какое смысловое значение имеет описание каждой из команд программы и какие действия должен выполнить компьютер при выполнении каждой из таких команд.

Интересно

Семантика — раздел языкознания, изучающий значения слов.

Например, чтобы получить сообщение 

, которое вы создавали в учебной среде *Скретч* с помощью команды 

, на языке программирования *Python*, нужно ввести команду: `print('Привет!')`. В алфавит этого языка входят латинские буквы и специальные символы для записи команд, цифры — для числовых данных, буквы русского или украинского алфавита — для комментариев и текстовых данных. Слово `print` входит в словарь языка и задаёт команду *Печатать*. Чтобы напечатать нужный текст, его записывают в скобках между символами «'» — таковы правила синтаксиса.

Для выполнения команд программы с помощью компьютера следует соблюдать все правила выбранного языка программирования. Если при составлении программы использовать символы, которые не входят в алфавит выбранного языка программирования, неправильно написать специальные слова или составить из этих слов структуру алгоритма, не соблюдая принятых правил, это будет воспринято как ошибка и программа не будет выполнена.

2. Чем отличаются языки программирования?

Сначала команды программы для исполнения компьютером писали с помощью обычных двоичных кодов. Программа выглядела очень громоздко, её составление занимало много времени. Со временем такие двоичные коды заменялись определёнными обозначениями, более понятными для человека. При этом создавались правила, а затем и программы, автоматически осуществлявшие перевод записанных таким образом программ для компьютера на язык машинных кодов. Сегодня существует свыше 3000 разнообразных языков программирования: некоторые из них уже не используются, а другие, напротив, совершенствуются и дают толчок к развитию новых языков и их версий.

Языки программирования можно разделить на две группы: языки **низкого уровня** и языки **высокого уровня** (рис. 15.1).

Машинный язык — набор команд, выполняемых непосредственно центральным процессором. Работать программисту с такими программами сложно из-за большого количества команд, записанных в двоичной форме. С момента создания компьютеров машинные коды были основным средством программирования.

К языкам низкого уровня относится **язык ассемблера** (от англ. *assemble* — составлять, компоновать). В языке ассемблера используются символьные обозначения команд, которые программисту легче понять и запомнить (рис. 15.2).

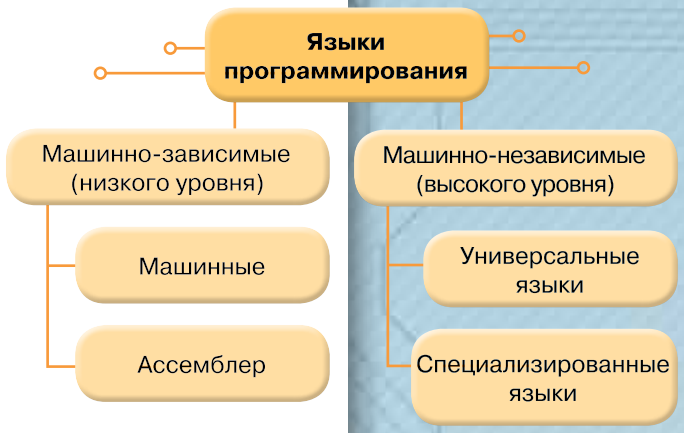


Рис. 15.1

Программа, записанная на языке программирования высокого уровня, содержит команды, похожие на обычные слова на английском языке. Например, программа, записанная на языке программирования *Python,* содержит команды `print` — печатать, `while` — пока, `if` — если и т. п.

Программы, записанные на языках высокого уровня, позволяют формулировать задания для выполнения на компьютере в привычном и понятном для человека виде, и именно поэтому использование компьютеров стало доступным широкому кругу людей, не являющихся специалистами в области программирования.

Различают универсальные и специализированные языки программирования высокого уровня. Универсальные языки используются для решения разных задач. К ним относятся *C++*, *C#*, *Pascal*, *Delphi*, *Java*, *C*, *Objective-C*, *Basic*, *Swift*, *Python*, *Cobol*, *D*, *Ada*. Специализированные языки предназначены для решения одного, максимум нескольких видов задач. Например, для работы с базами данных, веб-программирования или написания скриптов

Машинный код		Ассемблер	
005	B4 09	7	mov AH, 09h
0007	BA0000r	8	mov DX, offset msg
00A	CD21	9	int 21h

Рис. 15.2

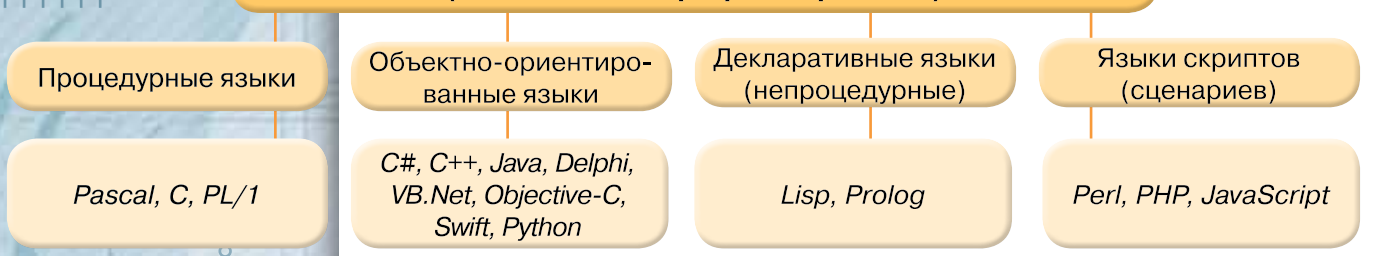


Рис. 15.3

для администрирования операционных систем. Примеры специализированных языков программирования — *Visual FoxPro*, *PHP*, *Perl*, *VBScript*, *JavaScript*, *VBA* в *Microsoft Office* и др.

По технологии программирования различают (рис. 15.3):

- **процедурные языки** — программа состоит из набора подзадач (процедур), реализующих задачу;
- **объектно-ориентированные** — главными элементами алгоритма является **класс** — новый тип данных, расширяющих язык, и **объект** с определёнными свойствами и методами — действиями, выполняемыми объектом;
- **декларативные** — языки создания программ с искусственным интеллектом;
- **языки скриптов** (языки сценариев) — языки, разработанные для описания «сценариев» — последовательностей команд, которые пользователь может выполнять на компьютере.

Интересно

Скриптовый язык (англ. *scripting language*) — язык программирования, разработанный для описания «сценариев», последовательностей операций, которые пользователь может выполнять на компьютере. В учебной среде создания и выполнения алгоритмов *Скретч* программы создают в области, которая называется *Скрипты*, потому что они реализуют сценарий событий, которые должны произойти с объектом-исполнителем.

ДЕЙСТВУЕМ

Упражнение 1. Знакомство с языками программирования.

Задание. Реализуйте в учебной среде создания программ *Блокли* (*https://blockly-demo.appspot.com/static/demos/index.html*) алгоритм, по которому в окне сообщения появится текст *Я изучаю языки программирования.*

1. Откройте окно браузера. Введите в поле адреса *https://blockly-demo.appspot.com/static/demos/index.html.*
2. На главной странице *Блокли* выберите среду *Code Editor* (рис. 15.4).

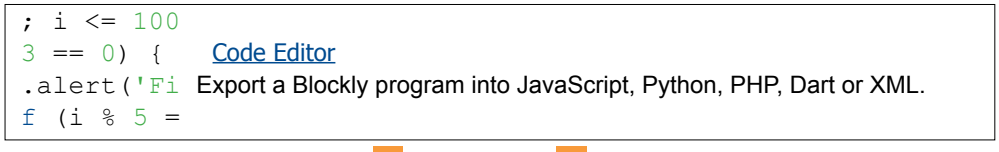


Рис. 15.4

3. В списке выбора языка среды English, расположенном в верхнем левом углу веб-страницы, выберите русский язык.
4. Составьте программу из блоков группы *Текст* (рис. 15.5) аналогично тому, как вы делали это в среде *Скретч.*

Рис. 15.5

5. Запустите программу на исполнение, нажав в левом верхнем углу окна среды кнопку *Запуск* . Проверьте, отображается ли в окне сообщения текст *Я изучаю языки программирования*.

6. Перейдите на страницу каждого из предложенных языков программирования (рис. 15.6).

JavaScript | Python | PHP | Dart | XML

Рис. 15.6

Ознакомьтесь с текстом программы на каждом из языков программирования. Определите, какая команда соответствует использованным блокам

.

7. Закройте окно браузера.

3. Какие средства необходимы для выполнения созданных программ?

Для преобразования команд языка программирования, отличающегося от машинного языка, используют специальные программы — **трансляторы**.

Транслятор (от англ. *translation* — перевод) — программа, записывающая на машинном языке команды программы, описанной на одном из языков программирования.

Различают два способа трансляции: компиляция и интерпретация.

Компилятор считывает сразу всю программу и переписывает её в машинном коде или на языке ассемблера. Процесс трансляции при таком подходе называется **компиляцией** и происходит один раз, а результат перевода хранится в отдельном файле. Если код программы изменяется, её необходимо будет перекомпилировать. Скомпилированная программа привязывается к операционной системе и набору команд процессора, поэтому не всегда может быть перенесена и выполнена на другом компьютере. С другой стороны, она является «готовой к использованию» и может быть быстро выполнена на том же или аналогичном компьютере: с точки зрения пользователя — просто щёлкнуть на имени исполняемого файла и запустить на выполнение, «с точки зрения» компьютера — просто «просчитать» и выполнить набор команд.

Интерпретатор считывает исходный код программы по одной команде и сразу пытается их «переводить» и выполнять. Это позволяет программисту быстрее проверять правильность выполнения программы и находить ошибки в коде. Выполнение программы при применении интерпретатора требует немного больше времени, поскольку каждый раз при запуске программы на выполнение происходит анализ кода и его преобразование с самого начала программы.

Поэтому для повышения быстродействия большинство современных интерпретаторов работают по смешанной схеме, сначала транслируя исходный код программы в некоторую промежуточную форму — так называемый байт-код. Это позволяет, при отсутствии изменений в оригинальной программе, не перечитывать её полностью, а использовать байт-код как «полуфабрикат» для работы. Выполнение байт-кода всё равно происходит медленнее выполнения машинного кода, но такой подход является компромиссом, сочетающим преимущества интерпретации и компиляции. Такой способ интерпретации используется в языке программирования *Python*, а также *Java* и *C#*.

Для создания программ, редактирования, поиска ошибок и их исправления, а также выполнения программ, написанных на языке программирования, используют среду программирования.

Интересно

Создание языка *Python* (произносят «пайтон») начал в конце 1980-х годов сотрудник голландского института CWI Гвидо ван Россум. Название языка не связано с известным пресмыкающимся, хотя для него и используют значок  . *Python* поддерживает высокоуровневые структуры данных и простой, но эффективный подход к объектно-ориентированному программированию. Элегантный синтаксис *Python*, динамическая типизация, а также то, что это интерпретированный язык, делают его идеальным для написания скриптов и быстрой разработки прикладных программ во многих отраслях.

Среда программирования — это комплекс программ, содержащий средства автоматизации процессов подготовки и выполнения программ пользователя, а именно:

- редактор кода программы — в нём можно создавать и редактировать текст программы;
- справочно-информационную систему о языке программирования и среде;
- библиотеки, в которых хранятся наиболее употребляемые фрагменты программ или целые программы;
- компилятор или интерпретатор, использование которого позволяет быстро найти в программе ошибку и исправить её;
- пошаговый «исполнитель» программы.

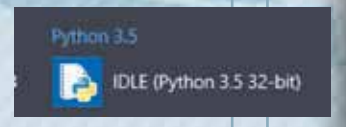
4. Какие среды программирования используются для создания программ?

Различают такие среды программирования: учебная, интегрированная, визуальная (рис. 15.7).

Для одного и того же языка программирования может существовать несколько сред программирования, поддерживающих разные технологии программирования. Они могут быть установлены на компьютер или реализованы в виде онлайн-сервиса.

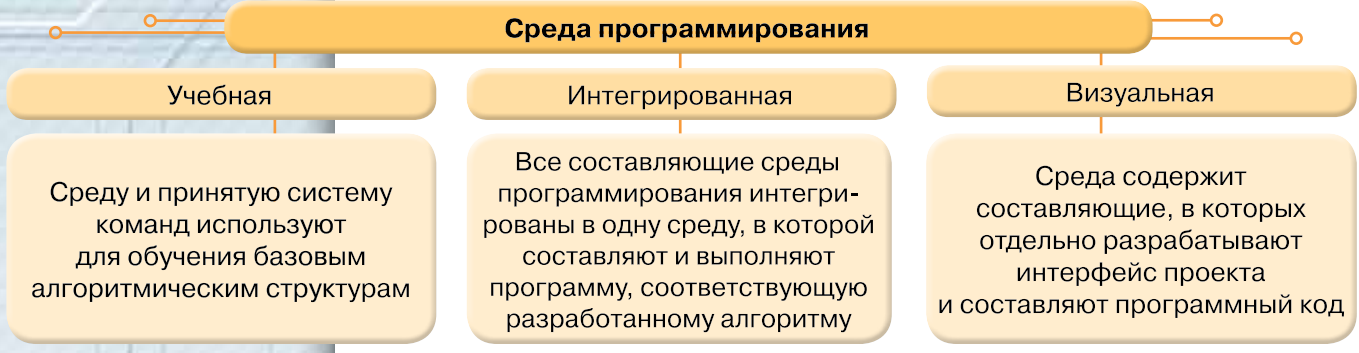
Среды *Скретч* и *Блокли* относятся к учебным средам программирования.

При установке на компьютер интерпретатора языка программирования *Python* одновременно также устанавливается интегрированная среда программирования *IDLE*, которую можно открыть из *Главного меню*. Особенностью этой среды является то, что при работе с ней используются два окна, в одном из которых записывается и редактируется текст программы, а в другом отображается результат её выполнения. После запуска программы *IDLE* по умолчанию открывается окно интерактивной среды интерпретатора *Python Shell*, в котором отображаются результат выполнения программы и сообщения об ошибках (рис. 15.8).



Если в этом окне в меню *File* (*Файл*) выбрать команду *New File* (*Новый файл*), то откроется окно редактора кода программы, в котором можно создавать, хранить, просматривать, редактировать, запускать программу, записанную на языке программирования *Python*. Например, чтобы отобразить на экране результат вычисления значения выражения *56 : 8 – 5,* достаточно в окне редактора кода программы использовать команду `print ()`, где в скобках записать выражение с использованием знаков арифметических операций: + (сложение), – (вычитание), * (умножение) и / (деление). Результат выполнения такого фрагмента

Рис. 15.7



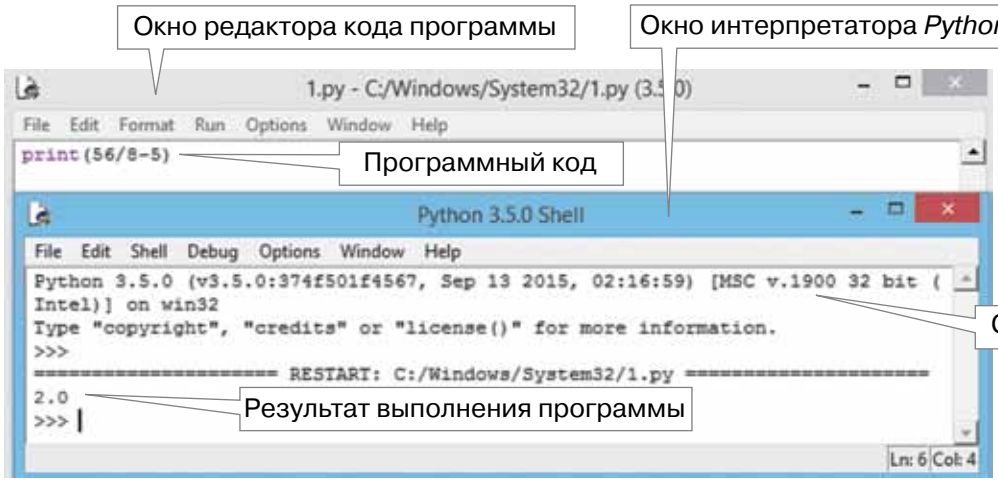


Рис. 15.8

программы после его сохранения и запуска с помощью команды *Run* (*Выполнить*) будет отображён в окне *Python Shell* (рис. 15.8).

Для создания и выполнения программ, записанных на языке программирования *Python,* можно использовать онлайновую интегрированную среду программирования *CodingGround* (*http://www.tutorialspoint.com/ipython_terminal_online.php*). После анализа системы на экране появляется номер строки, в которой можно вводить команды на языке *Python*. Например, на рисунке 15.9 приведён фрагмент программы для вычисления значения выражения, рассмотренного выше.

Удобной для использования интегрированной средой программирования, поддерживающей современные составляющие языка *Python,* является *PyCharm*. Эта среда совмещает редактор для ввода и редактирования программы, транслятор и отладчик ошибок. Пользователь получает возможность использовать статический анализ кода, подсветку синтаксиса и ошибок, навигацию среди проектов и кода программы, отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и многое другое.

Окно среды состоит из строки заголовка, меню, панели инструментов, панели навигации, структуры проекта, редактора кода программы, области

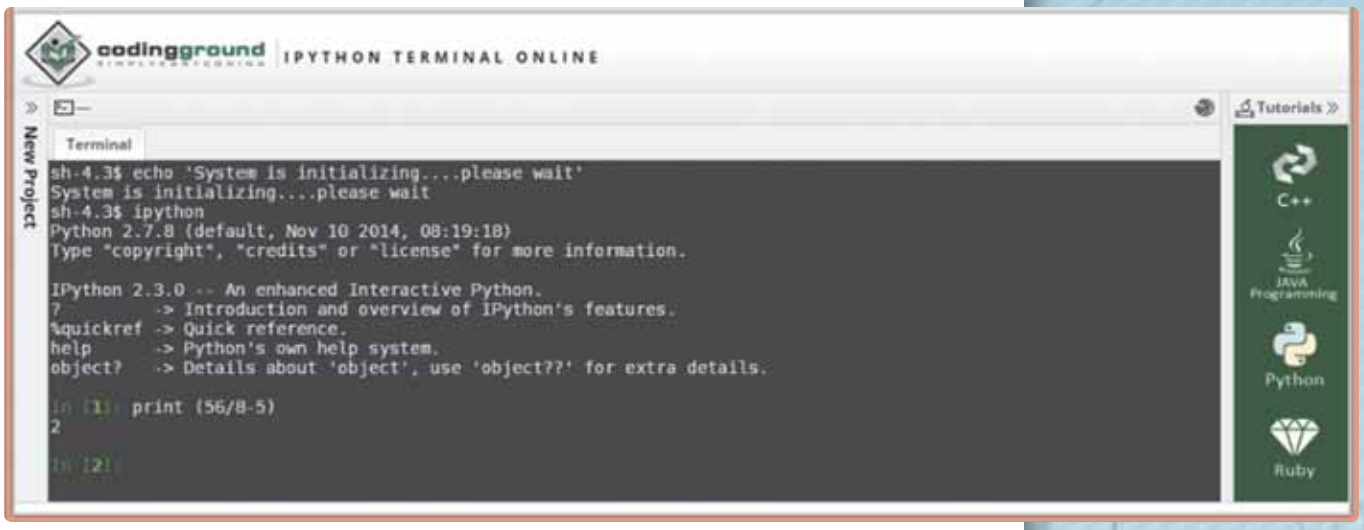


Рис. 15.9

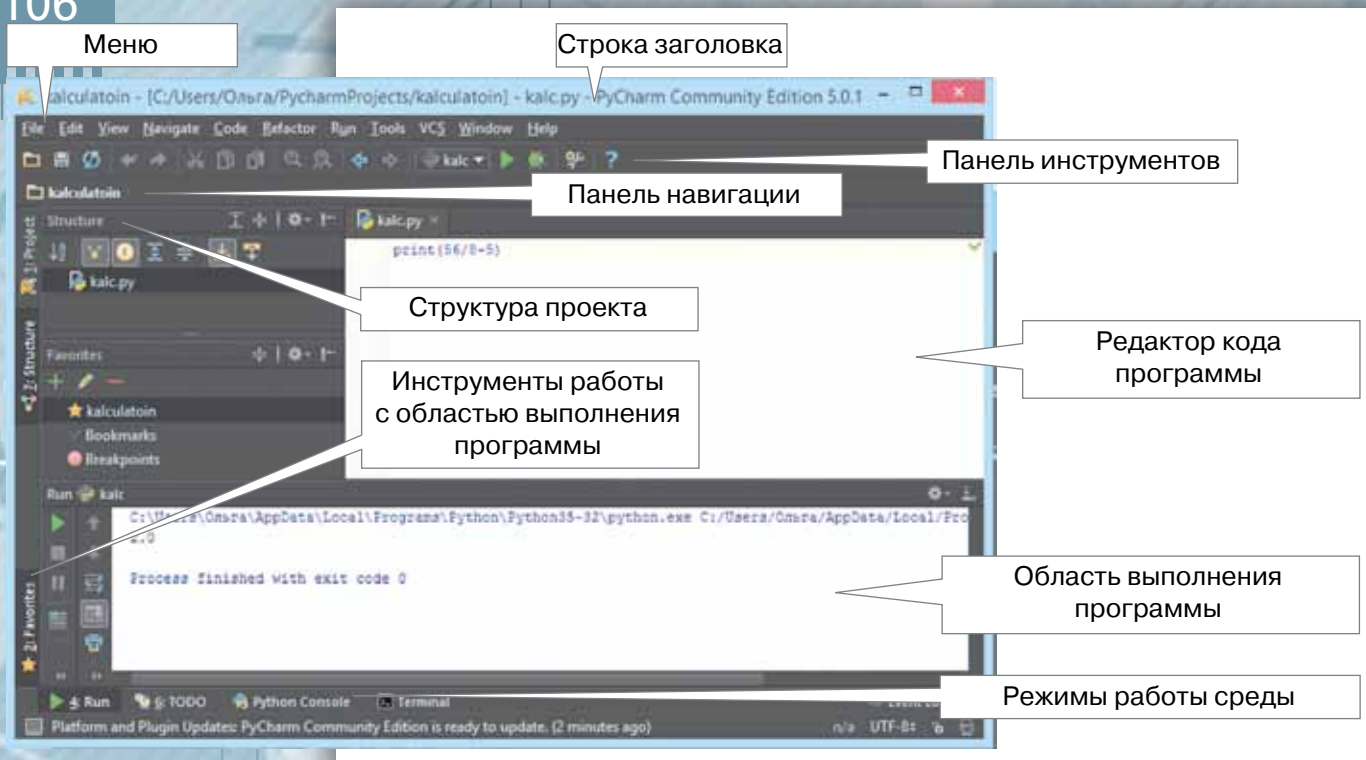


Рис. 15.10

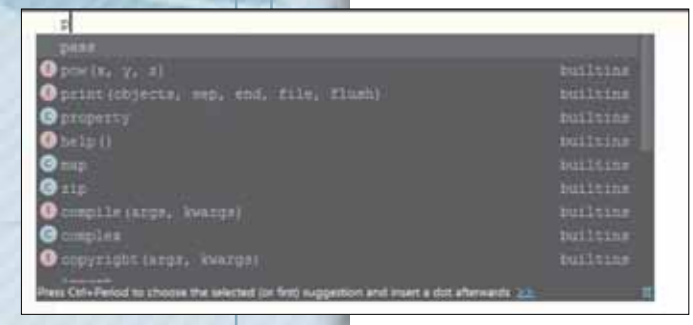


Рис. 15.11

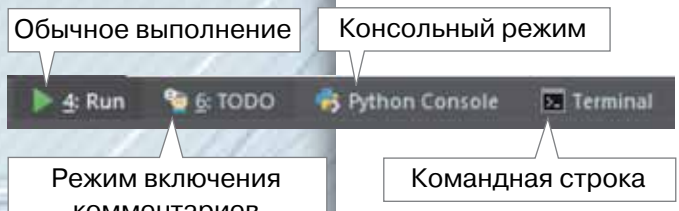


Рис. 15.12

выполнения программы, инструментов работы с областью выполнения программы, инструментов изменения режима работы среды (рис. 15.10).

Интегрированная среда программирования *PyCharm* содержит набор средств для помощи и поддержки пользователя. После загрузки среды появляется информационное окно, в котором можно узнать о полезных сервисах при работе с программой и комбинациях клавиш, с помощью которых они вызываются. Кроме этого, при вводе команды в редакторе кода программы сразу после ввода с клавиатуры первой буквы команды становится доступным список служебных команд с подсказками об их синтаксисе (рис. 15.11).

Область выполнения программы поддерживает несколько режимов, которые можно изменять с помощью соответствующих инструментов: обычное выполнение, режим включения комментариев, консольный режим, напоминающий среду *IDLE*, командная строка (рис. 15.12).

Для упрощения разработки графического интерфейса пользователя используют среды для визуального программирования, позволяющие конструировать программы, используя графические объекты. Для языка *Python* разработаны специальные библиотеки, позволяющие визуализировать процесс разработки проекта. Визуальные компоненты графического интерфейса проекта можно также создавать вручную, используя в программе соответствующие команды.

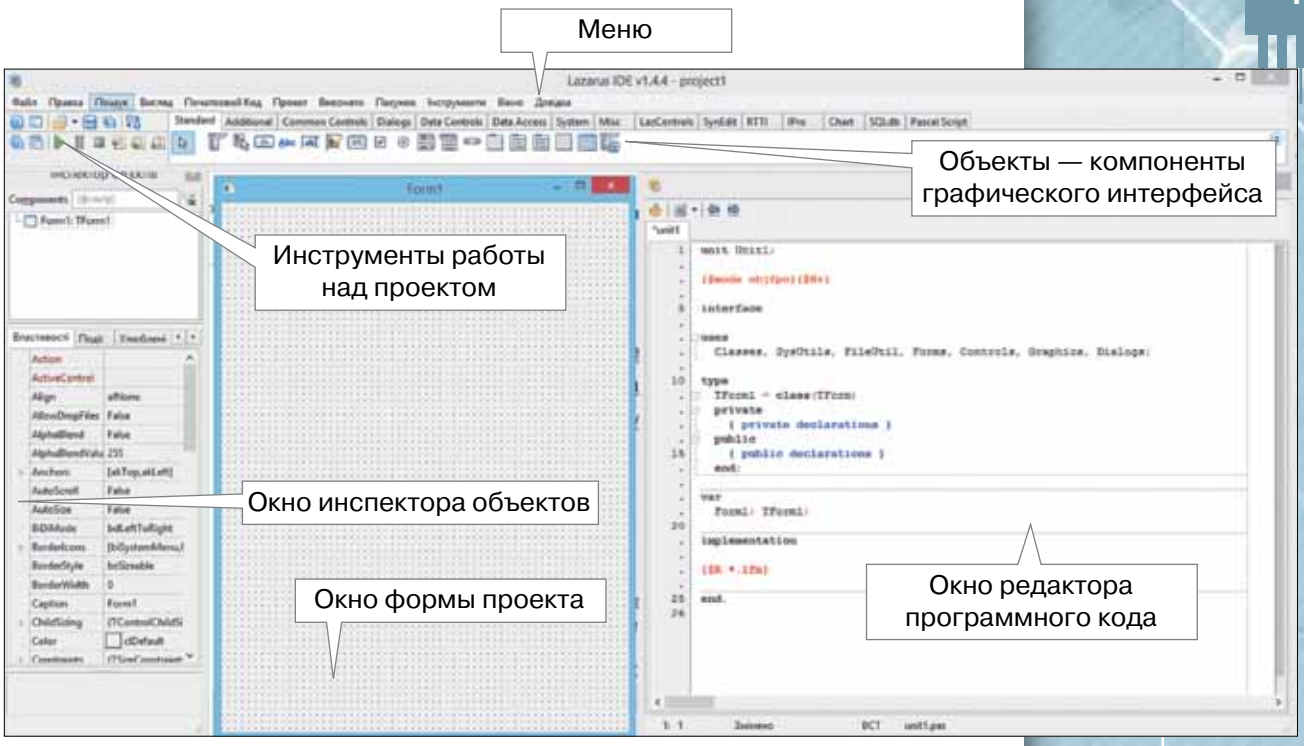


Рис. 15.13

Примером визуальной среды программирования является среда *Lazarus*, поддерживающая язык программирования *Free Pascal* (рис. 15.13). Для визуализации интерфейса проекта, созданного в этой среде, существует целый ряд специально разработанных элементов интерфейса — визуальных компонентов, позволяющих отображать данные разных типов — осуществлять управление программой в целом. Самый простой пример — кнопка в окне программы. Объект *кнопка* имитирует поведение обычной кнопки на пульте управления любого прибора. Её можно «нажимать» с помощью мыши. Окно формы проекта, в котором проектируется интерфейс, и окно, в котором создаётся программа, — разделены.

Такая среда похожа на среду *Скретч*, в области скриптов которой составляют программу, а на сцене размещают объекты, с которыми после запуска проекта на выполнение происходят события, соответствующие командам программы.

5. Как работать с проектом в среде программирования *PyCharm?*

В среде программирования *PyCharm* можно создавать несколько разных объектов:

- файл, который можно использовать, например, для хранения данных 

;
- папку, где можно сохранять, например, несколько проектов или вспомогательных программ, которые потом можно использовать в качестве файлов-заготовок для разработки новых проектов 

;
- пакеты 

;
- файл программы на языке *Python* 

;

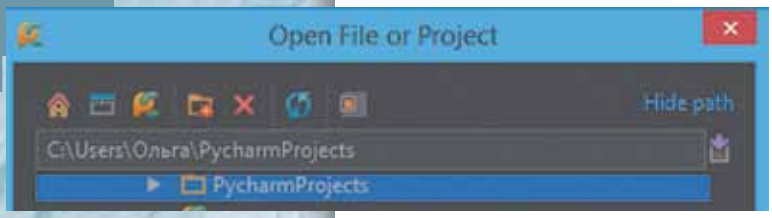


Рис. 15.14

- скрипт (программу) или любую страницу, которую можно разместить в Интернете HTML File.

Все эти объекты и их группы можно объединить в одном проекте, созданном с помощью команды *New Project* (*Новый проект*) из меню *File* (*Файл*).

Файл программы на языке *Python* имеет расширение *py*. Чтобы его открыть в среде *PyCharm*, используют команду *Open* (*Открыть*) из меню *File* (*Файл*). В окне открытия файла или проекта нужно указать путь к искомому файлу и нажать кнопку *OK* (рис. 15.14).

Чтобы выполнить программы, записанные в файлах проекта, используют команду *Run* (*Выполнить*) на панели инструментов или из одноимённого меню.

ДЕЙСТВУЕМ

Упражнение 2. Создание проекта в среде программирования *PyCharm*.
Задание. Создайте проект *Диалог* в среде *PyCharm*. Составьте программу на языке *Python* для ввода и вывода своего имени и фамилии.

1. Загрузите среду *PyCharm* одним из известных способов запуска программы.
2. Ознакомьтесь с предложенной справкой и закройте окно *Tip of the day* (*Совет дня*).
3. В меню *File* (*Файл*) выберите команду *New Project* (*Новый проект*).
4. В диалоговом окне *Create Project* (*Создание проекта*) выберите папку для размещения проекта и вместо имени по умолчанию *untitled1* задайте собственное имя — *dialog* (рис. 15.15). Нажмите кнопку *Create* (*Создать*).

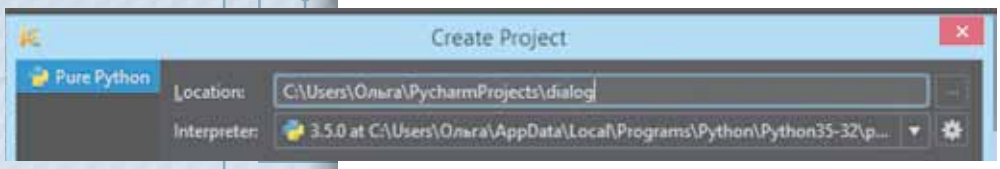


Рис. 15.15

Рис. 15.16

5. Создайте новый файл в проекте *dialog*. Для этого в меню *File* (*Файл*) выберите команду *New* (*Новый*), а в списке объектов — Python File. В окне *New Python file* (*Новый файл программы*) задайте имя файла *Привет*. Нажмите кнопку *OK* (рис. 15.16).
6. Введите текст программы в окне редактора кода (рис. 15.17). Обратите внимание: если текстовое сообщение на украинском языке содержит символ апостроф ('), то перед ним следует записать символ «\». В этом случае апостроф будет восприниматься как часть текста, а не указывать на окончание текстового фрагмента.

Строка, начинающаяся символом #, — это **комментарий** в тексте программы. Комментарии используют, чтобы помочь тому, кто знакомится с текстом программы. Они не влияют на ход выполнения программы.

```
# Запиши текст программы по образцу
print('Привет!')
print('Как тебя зовут?')
name = input('Введи своё имя)
print('Рад тебя приветствовать,', name)
```

Рис. 15.17

7. Запустите проект на выполнение. Для этого выберите инструмент *Run* (*Выполнить*) .
8. Обратите внимание на нижнюю часть окна среды и введите данные, необходимые для выполнения программы (рис.15.18).

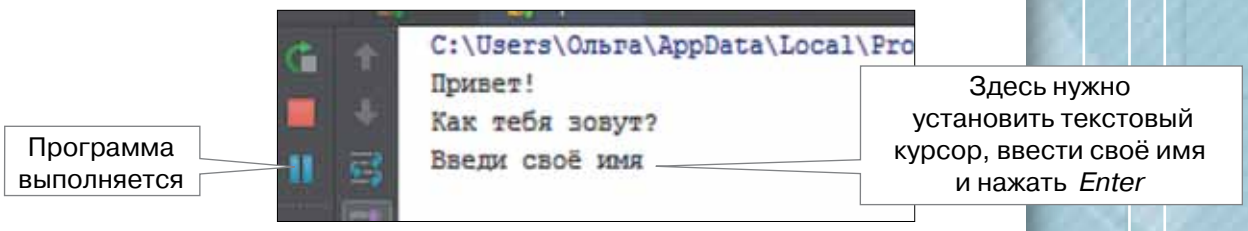


Рис. 15.18

9. Проверьте, завершилось ли выполнение всех команд — на экране должно появиться сообщение `Process finished with exit code 0`.
10. Очистите область выполнения программы. Для этого выберите инструмент — *Clear All* (*Очистить всё*).
11. Добавьте к программе команды так, чтобы после её запуска получить сообщения, которые, например, получил Вадим Антоненко (рис. 15.19).
12. Сохраните изменения в проекте. Для этого примените инструмент — *Save* (*Сохранить)* на панели инструментов.
13. Закройте окно среды. Для этого в меню *File* (*Файл*) выберите команду *Exit* (*Выйти*). В окне подтверждения выхода нажмите кнопку *Exit* (*Выйти*) (рис. 15.20).
14. Подтвердите завершение работы с программой *Привет* — выберите кнопку Disconnect.

```
Привет!
Как тебя зовут?
Введи своё имя Вадим
Рад тебя приветствовать,  Вадим
Какая твоя фамилия?
Введи свою фамилию Антоненко
Будем знакомы,  Вадим  Антоненко

Process finished with exit code 0
```

Рис. 15.19

Рис. 15.20

ИССЛЕДУЕМ

Упражнение 3. Ошибки в программе на языке *Python*.

Задание. Откройте проект *Dialog в среде программирования PyCharm*. Измените текст программы в файле *Привет*, как показано на рисунке 15.21. Проверьте, как в среде программирования сообщают пользователю об ошибке и как её исправление влияет на результат. Сделайте выводы.

Отсутствует символ « ' »

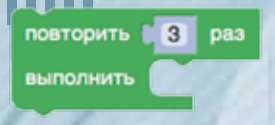
```
print('Привет!')
print('Как тебя зовут?')
name = nput('Введи
своё имя')
```

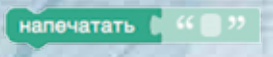
Ошибка в записи команды

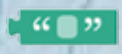
Рис. 15.21

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 15* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.







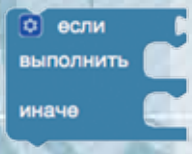
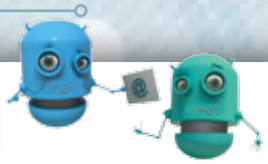
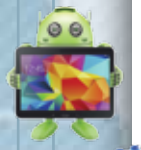


Рис. 15.22



РАБОТАЕМ В ПАРАХ

1. В среде *Блокли* создайте программу, которая в окне сообщения трижды будет выводить на экран некоторый текст. Обсудите, какие из предложенных блоков (рис. 15.22) можно для этого использовать.
Каким командам в среде *Скретч* соответствуют выбранные блоки? Определите, как команды повторения и печати реализуются в разных языках программирования.
2. Обсудите, что общего и чем отличаются учебные среды программирования *Скретч* и *Блокли*. Результат обсуждения представьте в виде диаграммы *Венна*.
3. Найдите в Интернете сведения об истории возникновения разных языков программирования. Обсудите и спланируйте, как можно было бы представить найденные сведения на временно́й шкале и в какой программе удобно было бы создать такую временную шкалу. Создайте её.
4. Обсудите, в каких случаях вы бы посоветовали своим одноклассникам для изучения языка *Python* использовать среду программирования *IDLE*, онлайновую среду *CodingGround*, интегрированную среду программирования *PyCharm*. Приведите по меньшей мере три аргумента для подтверждения своих предложений.



РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Определите, что общего и чем отличаются учебная среда программирования *Скретч* и среда визуального программирования *Lazarus*. Используйте инструменты текстового процессора, чтобы наглядно представить свой ответ.
2. Составьте на языке программирования *Python* программу, с помощью которой можно получить результаты вычисления значений выражений:
   1) $125 : 25 + 10$;
   2) $2 \cdot 3 \cdot 4 \cdot 5 : 6$;
   3) $765 : 9 + 48 \cdot 4 - 121$.

   Найдите значения указанных выражений. Сравните полученные результаты вычисления «вручную» и с помощью составленной программы. Сделайте выводы о правилах записи выражений, в которых предусмотрено использование арифметических операций, на языке программирования *Python*.
3. Откройте в среде программирования *PyCharm* программу *Guess.py,* находящуюся в папке *Программирование*. Запустите её на выполнение. Введите данные, соответствующие запросам программы. Определите, какое задание реализовано в этой программе.
4. Составьте программу *Терминал* на языке программирования *Python*, в которой получают сообщение о пополнении счёта оператора мобильной связи. Например:
   Введите название оператора мобильной связи Киевстар
   Введите номер телефона для пополнения счёта 0970707007
   Укажите сумму пополнения счёта 20
   Уважаемый абонент Киевстар! Вы пополняете телефон по номеру 0970707007 на сумму 20 грн.

16. ОБЪЕКТЫ ПРОГРАММ С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ

ВСПОМНИТЕ:

- что такое объект и чем отличаются объекты;
- в чём особенности графического интерфейса программы;
- какие объекты графического интерфейса используют в операционной системе *Windows*;
- как создают проекты в среде *Скретч*.

ВЫ УЗНАЕТЕ:

- каковы особенности программ с графическим интерфейсом;
- каковы способы создания объектов графического интерфейса;
- как работать с проектами в среде программирования *Lazarus*;
- как работать с экранной формой в среде *Lazarus*;
- как в среде *Lazarus* разработать прикладную программу.

ИЗУЧАЕМ

1. Каковы особенности программ с графическим интерфейсом?

Вы уже работали с разными программами, с графическим интерфейсом: редакторами, тренажёрами, программами для выполнения вычислений, играми и т. п. Они разработаны с помощью определённых сред программирования и имеют общие свойства:

- программа открывается в окне, размер которого, как правило, можно изменять;
- графический интерфейс программы, содержащий изображения значков, меню, кнопки, текстовые поля и т. п., позволяет пользователю с помощью мыши выполнять определённые команды, изменять значения свойств объектов и вводить с помощью клавиатуры текстовые и числовые данные;
- свойства всех использованных в программе объектов можно изменять;
- события, происходящие в программе, связаны с определёнными объектами. Например, с объектом *меню программы* связано событие, вызываемое нажатием левой кнопки мыши на выбранном разделе — развернуть список команд в этом разделе меню.

В основе разработки программ, использованных вами для решения различных прикладных задач, лежит **парадигма событийно-ориентированного программирования**: программа — это совокупность объектов реального или виртуального мира, имеющих определённые свойства и с каждым из которых связан некоторый набор событий (рис. 16.1).

Каждое событие приводит к некоторым очевидным или скрытым изменениям определённого объекта в соответствии с написанным программным кодом, создаваемым пользователем для каждого объекта отдельно. Программный код связан с окнами и элементами управления программной среды. Вам знакомы такие элементы управления окна программы, как кнопки, флажки, переключатели, списки, поля ввода, полосы прокрутки и т. п. Большинство из них предназначены для ввода данных пользователем и вывода результатов работы программы на экран (либо на бумагу, либо в файл), а также для управления работой компьютера.

Интересно

Парадигма программирования — это система идей и понятий, определяющих стиль написания программного кода, а также способ мышления программиста.

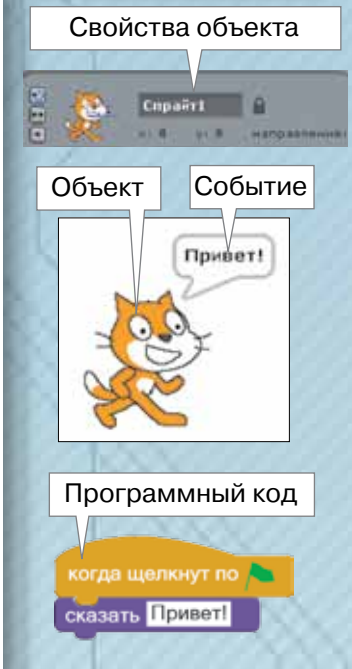


Рис. 16.1

В программном коде, ориентированном на обработку событий, программист должен указать, как реагировать на разные события (или действия пользователя). Это могут быть, например, такие события: выбор команды меню, щелчок кнопкой мыши, перемещение мыши и т. п.

2. Каковы способы создания объектов графического интерфейса?

Создавать объекты графического интерфейса в средах программирования можно двумя способами:

1) непосредственно в редакторе кода программы;
2) используя средства графического интерфейса среды визуального программирования.

Первый способ предусматривает подключение дополнительных программ — **модулей**, содержащих команды для создания графического интерфейса, которые можно использовать в программном коде. Создание объектов программы, их расположение и запуск обработки событий программы задаются командами языка программирования. Например, самая простая программа с графическим интерфейсом пользователя — вывод сообщения *Hello World!* в окне (рис. 16.2, *а*), созданная в среде программирования *PyCharm*, состоит из команд, представленных на рисунке 16.2, *б*.

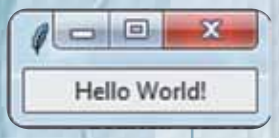


Рис. 16.2, *а*

```
# подключение модуля оконного графического интерфейса пользователя
import tkinter
# создание окна программы
main = tkinter.Tk()
# создание текстовой надписи
label = tkinter.Label(text="Hello World!")
# размещение надписи на главной форме
label.pack()
# запуск обработки событий программы
main.mainloop()
```

Рис. 16.2, *б*

В среде визуального программирования каждый программный проект состоит из интерфейса пользователя, обеспечивающего ввод и вывод данных, управление их обработкой, а также из программного кода — программы (рис. 16.3). Результатом выполнения программы с графическим интерфейсом является экранная форма, содержащая объекты. Программа может обрабатывать одну или несколько экранных форм.

Рис. 16.3

Интерфейсом пользователя в программе называется визуализированная часть программы, с помощью которой пользователю представляются сведения и принимаются от него данные для управления работой программы.

Существуют различные интерфейсы пользователя. В ОС *Windows* принят стандартный графический интерфейс, в котором пользователь использует мышь. Так, например, интерфейс пользователя программ для ОС *Windows* может состоять из меню, одной или нескольких панелей инструментов или лент (наборов кнопок с рисунками) для ускорения выполнения действий и рабочей области, имеющей вид документа или развёрнутого листа. К интерфейсу программ, работающих под управлением ОС *Windows*, относят такие основные объекты, как **окна** (прикладных программ, сообщений, документов и диалоговые), **панели инструментов** и **меню**, каждый из которых также имеет свои элементы. Их совокупность образует **визуальную составляющую** любой прикладной программы. Формировать такую визуальную составляющую можно с помощью специальных средств графического редактирования (компоновки), например, в среде программирования *Lazarus*.

ДЕЙСТВУЕМ

Упражнение 1. Окно сообщения.

Задание. Создайте программу в среде программирования *PyCharm*, с помощью которой в окне сообщения будет выводиться цитата академика В. М. Глушкова:

```
Человек в XXI веке, который не будет уметь пользоваться компью-
тером, будет подобен человеку XX века, не умевшему ни читать,
ни писать.
```

1. Запустите среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Текст* в папке *Учебные проекты* своей структуры папок.
3. Подключите к файлу модуль оконного графического интерфейса пользователя, создайте объекты: окно, две текстовые надписи, отображающие текст цитаты и её автора. Воспользуйтесь командами, как на рисунке 16.2, *б*, или загрузите заготовку фрагмента кода программы из файла *Окно* папки *Программирование*.
4. Разместите надпись и команду запуска обработки события программы на главной форме. Например, для вывода в окне сообщения текста цитаты нужно ввести такой программный код:

```
label1 = tkinter.Label(text="Человек в XXI веке, который
не будет уметь пользоваться компьютером, будет подобен
человеку XX века, не умевшему ни читать, ни писать.")
label1.pack()
```

5. Запишите команды создания надписи и запуска обработки события для вывода в окне автора высказывания.
6. Запустите файл программы на выполнение. Убедитесь, что на экране вы получили нужное окно с цитатой.
7. Завершите работу со средой программирования. Закройте все окна программ.

3. Как работать с проектами в среде программирования *Lazarus*?

Вы знаете, что файл, созданный в среде *Скретч*, называется **проектом**. Проект в *Скретч* хранится в одном файле. В нём содержатся программы для каждого из исполнителей, образы этих исполнителей, фоны сцены, звуки.

С проектами работают не только в учебных алгоритмических средах, их создают также и в средах программирования.

Проект — это набор файлов, с которыми пользователь работает при создании прикладной программы в объектно-ориентированной среде программирования.

В проект входят как стандартные файлы, так и файлы, создаваемые пользователем. В файлах хранятся отдельные экранные формы и модули. Каждый проект должен иметь имя.

Кроме того, проектом называется сама прикладная программа в процессе её создания.

Проект, созданный в среде *Lazarus*, в отличие от среды *Скретч*, содержит несколько файлов:

- *project1.exe*, с помощью которого запускают проект на выполнение;
- *unit1* для сохранения программного кода проекта, написанного на языке программирования *Free Pascal* (файл *unit1.pas*);
- *unit.lfm*, содержащий данные, полученные в процессе проектирования формы;
- другие файлы и папки.

Рис. 16.4

Папка проекта может содержать папки и файлы, в частности, как на рисунке 16.4.

В среде *Lazarus* можно создавать новые проекты или использовать шаблоны проектов, открывать созданные проекты, закрывать и сохранить их, изменять структуру составляющих проекта и т. п. Эти действия выполняют с помощью соответствующих команд меню *Проект* (рис. 16.5).

После выбора соответствующей команды из меню в открывающихся окнах указывают необходимые параметры и задают их значения. Например, с помощью команды *Проект/Создать проект* открывается окно, в котором указывают тип проекта: программа с графическим интерфейсом, простая программа и т. п. (рис. 16.6).

При загрузке среды *Lazarus* по умолчанию автоматически загружается последний проект, с которым работали в этой среде. Чтобы этого не происходило, в меню *Инструменты* нужно выбрать команду *Параметры*, в окне *Параметры IDE* — снять метку *Открывать последний проект при запуске* (рис. 16.7) и завершить процесс внесения изменений настроек нажатием кнопки *OK*.

Загруженный с носителей или вновь созданный проект можно выполнить. Для этого используют инструмент *Выполнить*  , расположенный на панели инструментов или в меню *Выполнить*.

После запуска проекта на выполнение в окне *Сообщения* отображаются сведения о ходе загрузки и компиляции проекта (рис. 16.8). Зелёный цвет сообщения свидетельствует об успешной компиляции проекта, жёлтый — о наличии ошибок, позволяющих выполнить проект, красный — об ошибках, не позволяющих завершить компиляцию проекта.

После компиляции проекта пользователь может работать с **экранной формой**, свернув или закрыв окно среды.

Экранная форма — это окно, содержащее визуальные графические элементы или объекты управления, такие как меню, кнопки. Каждый такой

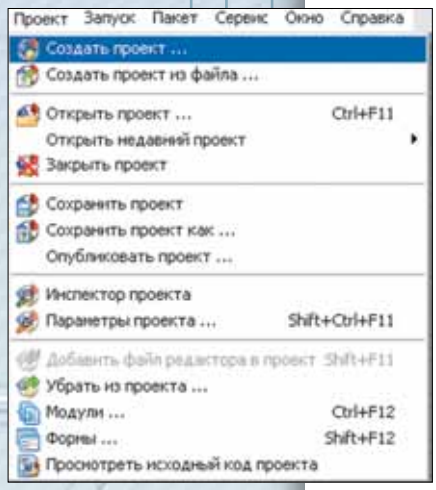


Рис. 16.5

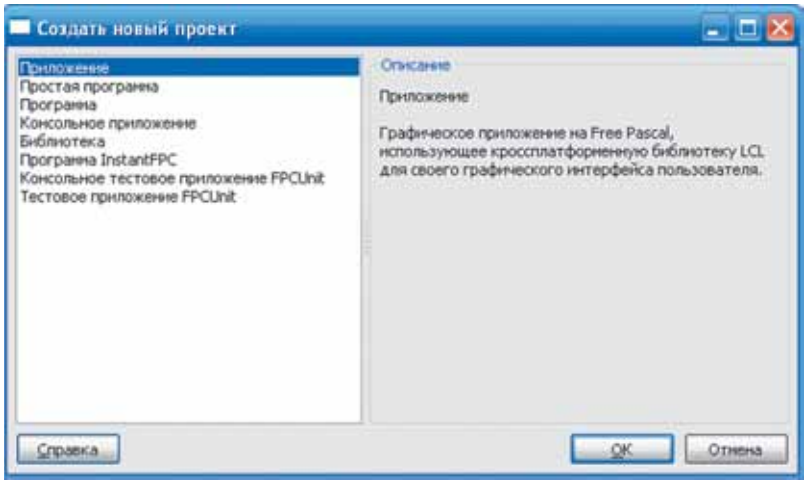


Рис. 16.6

Рис. 16.7

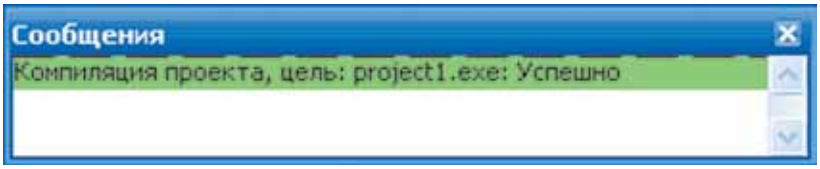


Рис. 16.8

объект может иметь связанный с ним функциональный программный код, оформленный в виде процедуры обработки события. Событийные процедуры можно разместить только в пределах программного кода, связанного с одной конкретной экранной формой. После запуска программы на выполнение для каждой экранной формы создаётся файл с расширением *lfm*, в котором хранятся данные, полученные в процессе проектирования формы.

Упражнение 2. Загрузка и выполнение проекта.

Задание. Загрузите и выполните проект *Масса тела*, находящийся в папке *Учебные проекты*. Введите данные, необходимые для выполнения проекта, и проанализируйте результат его выполнения.

1. Откройте среду программирования *Lazarus*.
2. В меню *Проект* выберите команду *Открыть проект.*
3. Перейдите к папке *Учебные проекты*. В папке проекта *Масса тела* откройте файл проекта *project1.lpi*.
4. Запустите проект на выполнение. Введите в предлагаемые поля свой рост в сантиметрах и массу в килограммах (рис. 16.9).
5. Нажмите кнопку *Рассчитать*, чтобы проверить показатели соотношения своего веса и роста. Проанализируйте полученный результат.
6. Закройте окно формы проекта. В окне сообщения о завершении выполнения проекта нажмите кнопку *OK* (рис. 16.10).
7. Завершите работу с проектом. Для этого в меню *Проект* выберите команду *Закрыть проект*.
8. Завершите работу со средой. Для этого в окне *Мастер проектов* выберите Выйти из Lazarus.

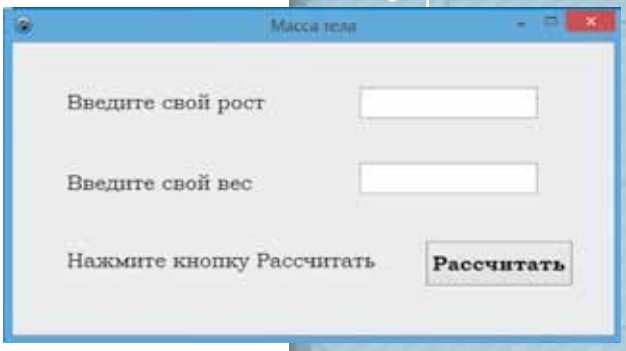


Рис. 16.9

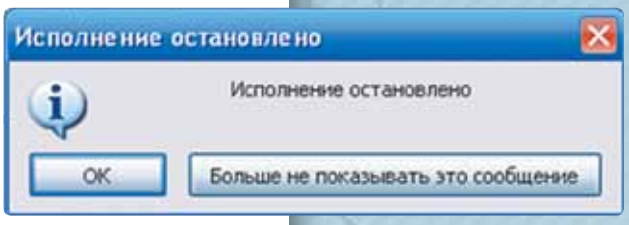


Рис. 16.10

4. Как работать с экранной формой в среде *Lazarus*?

Экранную форму можно создать в окне дизайнера формы проекта, содержащего совокупность созданных пользователем объектов. В то же время в окне инспектора объектов будет отображаться список свойств каждого объекта, созданного пользователем, и формы проекта.

В начале создания экранной формы окно дизайнера формы проекта выглядит как пустая рабочая область (рис. 16.11), на которой пользователь может размещать различные **объекты** или **элементы управления:** текстовые надписи,

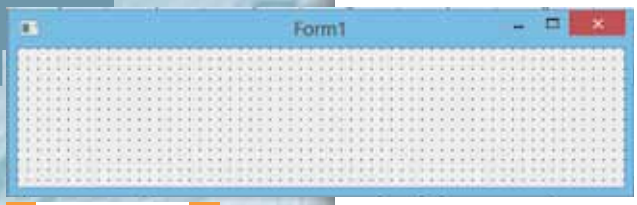


Рис. 16.11

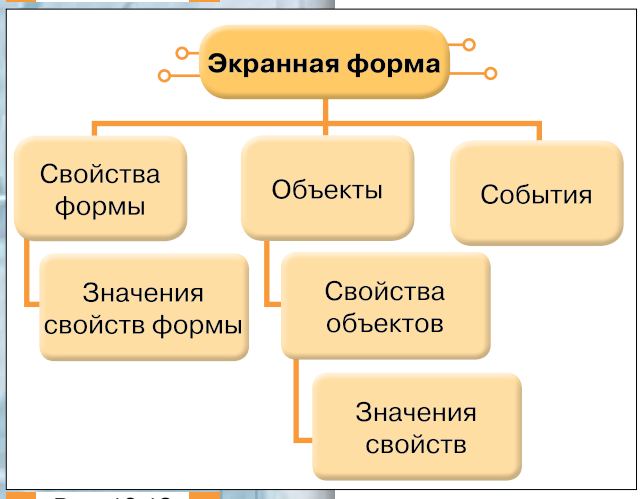


Рис. 16.12

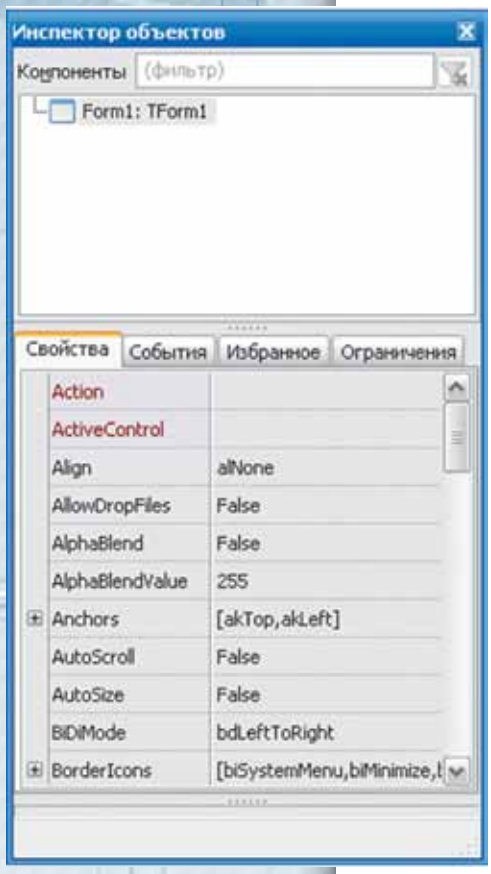


Рис. 16.13

кнопки, переключатели, рисунки и т. п., — подобно холсту для художника, на котором тот рисует предметы.

Экранная форма также является объектом, следовательно, имеет свойства и имя, она «может реагировать» на события из некоторого списка (рис. 16.12).

По умолчанию экранной форме присваивается имя `Form1`. При необходимости можно открыть дополнительные формы, у которых будут имена `Form2`, `Form3` и т. д. Все эти формы будут равноправными и независимыми одна от другой.

Добавление новой формы выполняется с помощью команды меню *Файл/Новая форма* или с помощью инструмента *Новая форма*  , расположенного на панели инструментов *Быстрые клавиши IDE*. Чтобы просмотреть список всех форм проекта, можно воспользоваться инструментом *Показывать формы*  , расположенным на той же панели инструментов.

Чтобы изменить значение свойств формы, например имя, используют таблицу вкладки *Свойства* окна *Инспектор объектов* (рис. 16.13).

Все свойства объектов разделяют на простые и составные. Простые свойства определены единственным значением. Например, свойство `Caption` (*Подпись*) определяется строкой символов, которые могут заменить имя формы `Form1`. Свойства `Height` (*Высота*) и `Width` (*Ширина*) — числовым значением. Свойство, определяющее, будет ли размер окна автоматически изменяться в соответствии с содержимым `AutoSize` (*Авторазмер*), — значениями `True` (*Истина*) и `False` (*Ложь*). Свойство `Position` (*Расположение*) выбирается из списка предложенных. Составные свойства определяются совокупностью значений. Например, у свойства `Font` (*Шрифт*) несколько параметров форматирования: начертание, размер, цвет символов и т. п., — значение которых можно изменить в окне *Шрифт* (рис. 16.14). Окно для изменения составных свойств открывается с помощью кнопки  , расположенной в правой части строки выбранного свойства.

5. Как в среде *Lazarus* разработать прикладную программу?

Разработка прикладной программы в среде *Lazarus* состоит из нескольких этапов:

1. Анализ и планирование выполнения задания.
2. Подготовка проекта.
3. Размещение компонентов интерфейса пользователя на форме проекта.
4. Написание программы обработки событий.
5. Тестирование и отладка программы.

На первом этапе, кроме определения исходных данных и результатов, необходимо построить информационную модель решения

задачи, определить методы решения и составить соответствующие алгоритмы.

На этапе подготовки проекта целесообразно создать для него отдельную папку. После запуска среды программирования *Lazarus* следует создать новый проект — программу с поддержкой графического интерфейса, сохранить его в созданной папке, проверить успешность компиляции и запуска нового пустого проекта — это можно сделать также, используя клавишу *F9*. Далее — закрыть окно запущенного проекта и продолжить его создание. Быстро сохранять все промежуточные изменения можно с помощью комбинации клавиш *Ctrl+S*.

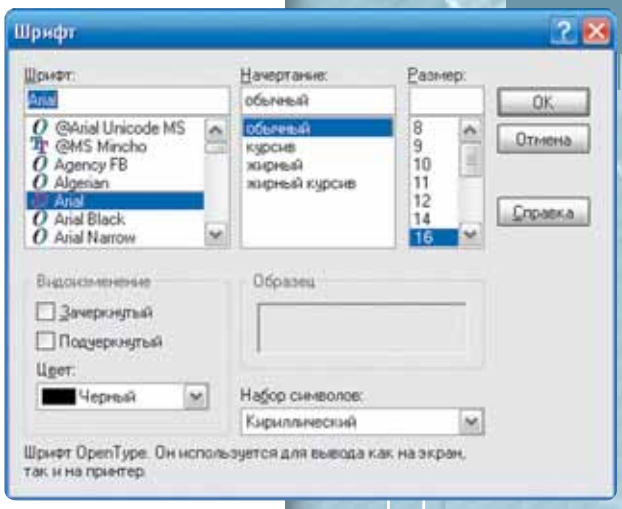


Рис. 16.14

При создании графического интерфейса программы следует разместить компоненты этого интерфейса на форме и установить значения их свойств. Чтобы разместить объекты на форме на одном уровне по горизонтали или вертикали, используют автовыравнивание, показывающее размещение одних компонентов относительно других по горизонтали и по вертикали с помощью линий выравнивания (рис. 16.15).

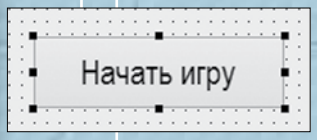


Рис. 16.15

Программы обработки событий создаются в окне редактора кода системой автоматически или составляются вручную средствами языка программирования *Free Pascal*.

Созданный проект запускают на выполнение. Если код программы не содержит синтаксических ошибок и программа успешно откомпилирована, отобразится экранная форма. Следует проверить, все ли объекты экранной формы «реагируют» на события так, как это предусмотрено заданием проекта. При наличии несоответствия поставленным задачам необходимо внести изменения в код программы или изменить значения свойств объектов.

ДЕЙСТВУЕМ

Упражнение 3. Создание проекта в среде *Lazarus*.

Задание. Создайте проект, после запуска которого в окне *Первая программа* выводится сообщение *Ура! Заработало!*.

1. Спланируйте проект. Подумайте, какой текст должен появляться в строке заголовка экранной формы, а какой — в её рабочей области. Нарисуйте модель размещения компонентов графического интерфейса.
2. Выполните шаги подготовительного этапа разработки проекта. Создайте папку *Первая программа*.
3. После запуска среды *Lazarus*, сохранения проекта и его компиляции перейдите в окно дизайнера формы.
4. В таблице свойств формы выберите свойство `Caption` (*Подпись*), задайте значение этого свойства — *Первая программа.*
5. На панели компонентов *Standart* (*Стандартная*) выберите инструмент Abc, с помощью которого создаётся надпись на форме.
6. Щёлкните в том месте формы, где вы планируете разместить объект *Надпись*.
7. В таблице свойств надписи выберите свойство `Caption` (*Подпись*), задайте значение этого свойства — *Ура! Заработало!.* Проверьте, изменилась ли надпись *Label1* на введённый текст.
8. Измените значение свойства `Font` (*Шрифт*) так, как на рисунке 16.14.

9. Сохраните изменения в проекте. Запустите его на выполнение. Сверните все окна среды *Lazarus*. При необходимости отладьте проект. Закройте все окна.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 16* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

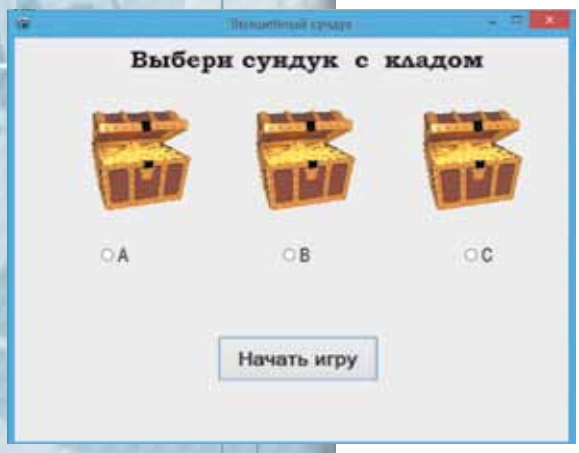


Рис. 16.16

1. Обсудите назначение основных файлов проекта (рис. 16.4). Подумайте, почему у некоторых файлов имя одинаковое, а расширение — разное. Каково их назначение в проекте, созданном в среде *Lazarus?*
2. Определите три аргумента, подтверждающие, что учебная среда программирования *Скретч* поддерживает событийно-ориентированное программирование.
3. Запустите на выполнение проект *Волшебный сундук*, хранящийся в папке *Проект Сундук* папки *Учебные проекты* (рис. 16.16). Обсудите, какие компоненты графического интерфейса могли быть использованы при его составлении, каковы значения их основных свойств. Убедитесь в правильности своих догадок, открыв проект в среде *Lazarus.*

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Ваш младший брат любит долго работать за компьютером. Создайте в среде *PyCharm* программу, после запуска которой будет появляться окно с сообщением о необходимости соблюдения времени работы за компьютером. Создайте аналогичную программу в среде *Lazarus*. Определите:
   1) сколько времени вы затратили на разработку каждой из программ, подумайте, почему зафиксированные значения времени разработки программы отличаются;
   2) окно какой экранной формы проекта больше заинтересовало бы младшего брата и почему.
2. Вы хотите использовать компьютер для планирования деятельности своей семьи. Выберите среду программирования, спланируйте и создайте прикладную программу, после запуска которой в окне будет появляться сообщение, актуальное для вашей семьи, например, напоминания о дне рождения, важном событии, запланированном задании и т. п.
3. Выберите среду программирования *Lazarus* или *PyCharm* и создайте в ней проект, после запуска которого будет появляться окно с описанием понятий этого урока: проект, экранная форма, интерфейс пользователя.

ПОЛЕЗНЫЕ ССЫЛКИ

Основы программирования на языке *Python*: *https://www.thinkful.com/learn/intro-to-python-tutorial/*

Вики-страница справочных материалов по среде *Lazarus*: *http://wiki.freepascal.org/Lazarus_Tutorial/ru*

4. Создайте программу, после запуска которой на экранной форме отобразится календарь на текущий месяц, например, на ноябрь 2016 г. (рис. 16.17). Воспользуйтесь подсказкой: компоненты графического интерфейса, размещённые на форме, можно копировать, например, с помощью команд контекстного меню. После вставки создаётся новый компонент, повторяющий свойства предыдущего, которые можно изменить.

Рис. 16.17

17. ПРАКТИЧЕСКАЯ РАБОТА 8

СОЗДАНИЕ ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ ПРОГРАММЫ, ОТОБРАЖАЮЩЕЙ ОКНО СООБЩЕНИЯ

ВСПОМНИТЕ

- Как размещать текстовые надписи на экранной форме проекта;
- как изменять свойства объектов программы;
- как создавать проекты в средах программирования *PyCharm* и *Lazarus*.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 8.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Сообщение о выполнении практической работы (*9 баллов*)
Составьте программу *Практическая_работа* в среде *PyCharm*, с помощью которой в окне будет выводиться сообщение о номере практической работы, дате её выполнения, фамилии и имени исполнителя по образцу (рис. 17.1).

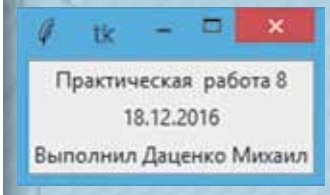


Рис. 17.1

Задание 2. Кислоты (*9 баллов*)
Распределите названия кислот в окне программного проекта *Кислоты*, находящегося в папке *Программирование*, на две группы. Измените значения свойств надписей так, чтобы значения параметров форматирования символов в названиях групп были одинаковыми.

Задание 3. Словосочетание (*12 баллов*)
Составьте программу, которая в окне с заголовком *Словосочетание* будет выводить напоминание о видах связи слов в словосочетаниях, известных из уроков русского языка (рис. 17.2).

Рис. 17.2

18. СВОЙСТВА И МЕТОДЫ ЭКРАННОЙ ФОРМЫ И ЭЛЕМЕНТОВ УПРАВЛЕНИЯ

ВСПОМНИТЕ:

- какие действия можно выполнить с элементом управления *кнопка* в программах с графическим интерфейсом;
- для чего используются кнопки управления окном в ОС *Windows*;
- как в среде *Скретч* можно создать объект *кнопка*;
- с помощью каких команд в среде *Скретч* можно передать действие одного объекта другому.

ВЫ УЗНАЕТЕ:

- как изменить значения свойств объектов в среде программирования *Lazarus*;
- как объекты могут «реагировать» на события;
- как в проекте используется элемент управления *кнопка*;
- как выполнять действия с объектами с помощью методов;
- как в средах программирования можно использовать окна сообщений.

ИЗУЧАЕМ

1. Как изменить значения свойств объектов в среде программирования *Lazarus*?

Все элементы управления, размещаемые на форме проекта, как и сама форма, являются объектами. Вы уже умеете устанавливать значения свойств объекта, в частности надписи и формы, **статическим** способом. Это значит, что значения свойств объектов устанавливаются до запуска программы на выполнение. Пользователь сначала выделяет в окне дизайнера формы объект, значение свойства которого необходимо изменить, а затем в окне *Инспектор объектов* в таблице свойств находит название соответствующего свойства и справа от этого названия выбирает или вводит с клавиатуры нужное значение.

Для изменения значений свойств можно применить и другой способ — **динамический**, когда значения свойств можно изменить в процессе выполнения программы с помощью команды изменения значений — **присваивания**. Для записи команды присваивания используют **оператор присваивания**. На языке программирования *Free Pascal*, поддерживаемом в среде *Lazarus*, такой оператор обозначают «:=», а на языке *Python* — просто «=». Слева от такого оператора записывается название свойства, а справа — значение, которое необходимо задать. Такие команды пишутся в программном коде, и значение свойства будет изменяться только после запуска программы на выполнение. Для того чтобы при написании программного кода указать, что значение свойства следует изменить именно для выбранного объекта, необходимо, кроме названия свойства в команде задания значения, ещё указать имя объекта, для которого эти изменения предусмотрены. Если проект состоит из нескольких экранных форм, следует указать также имя формы, на которой размещён такой объект. Поэтому для обращения к свойствам объектов в программном коде используют такой способ записи:

```
Имя формы.Имя объекта.Название свойства
```

Интересно

В среде программирования *PyCharm* используют динамический способ изменения свойств объектов. Так, с помощью команды

```
label = tkinter.Label (text="Hello World!")
```

не только создают надпись в окне, но и указывают текст, который будет отображаться в этой надписи при запуске программы на выполнение.

Если проект содержит только одну форму или рассматриваются объекты текущей формы, то указывать имя формы необязательно.

Так, например, чтобы в проекте, содержащем только одну форму, объект `Label1` (*Надпись1*) отображался на форме, необходимо его свойству `Visible` (*Видимость*) присвоить значение `True`. Для этого в коде программы записывают команду:

```
Label1.Visible := True;
```

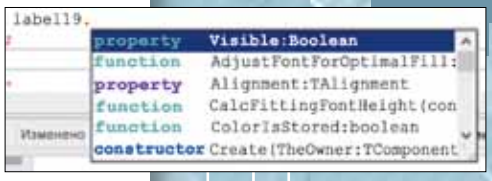


Рис. 18.1

Свойство `Visible` (*Видимость*) может принимать одно из двух значений: `True` — тогда объект будет отображаться, и `False` — в этом случае получим невидимый, или «скрытый» объект.

В окне редактора кода среды программирования *Lazarus* после ввода имени объекта и точки отображается список доступных свойств и действий, которые можно описать для объекта (рис. 18.1). Чтобы добавить в код название нужного свойства, достаточно выбрать его мышью из списка.

2. Как объекты могут «реагировать» на события?

При выполнении программы объекты могут «реагировать» на некоторые **события**, например действия пользователя: щелчок мышью на объекте, нажатие на клавиатуре какой-либо клавиши или комбинации клавиш, выбор некоторой команды меню, изменение размеров окна и т. п. Т. е., если происходит некоторое предусмотренное для объекта событие, выполняется определённый набор команд. Например, при работе с офисными программами при первом сохранении файла пользователь может выбрать на панели инструментов кнопку *Сохранить*  , или нажать на клавиатуре комбинацию клавиш *Ctrl* + *S*, или в меню *Файл* выбрать команду *Сохранить*. Если происходит любое из этих событий, будет выполнено действие: откроется окно *Сохранить как*, в котором можно выбрать имя файла и папку, где этот файл следует сохранить.

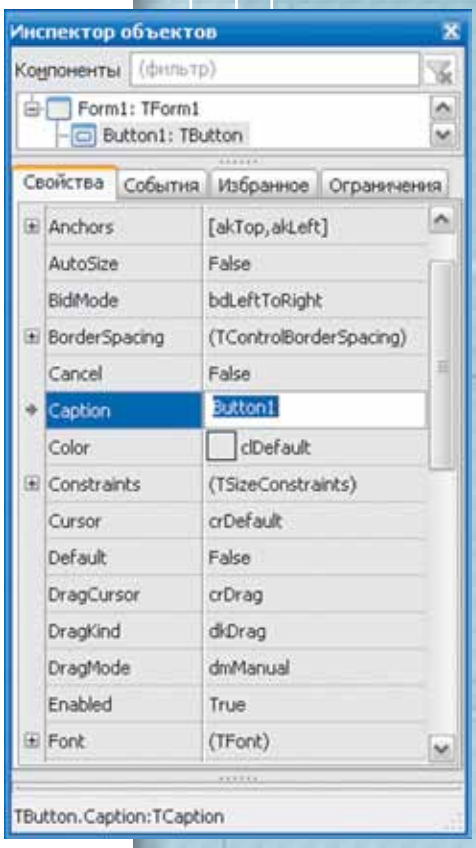


Рис. 18.2

Чтобы описать действия, которые должны выполняться вследствие некоторого происходящего для объекта события, необходимо написать программный код — подпрограмму, начинающуюся в среде *Lazarus* со служебного слова `procedure`.

Список всех доступных событий, которые могут обрабатываться, для каждого объекта в среде программирования *Lazarus* представлен в таблице на вкладке *События* окна *Инспектор объектов* (рис. 18.2).

Список событий, которые чаще всего применяются при составлении проектов с выбранным объектом, отображается в таблице на вкладке *Избранное*. Например, список таких событий для формы представлен на рисунке 18.3.

Если дважды щёлкнуть в ячейке таблицы справа от выбранного события, то в окне редактора кода

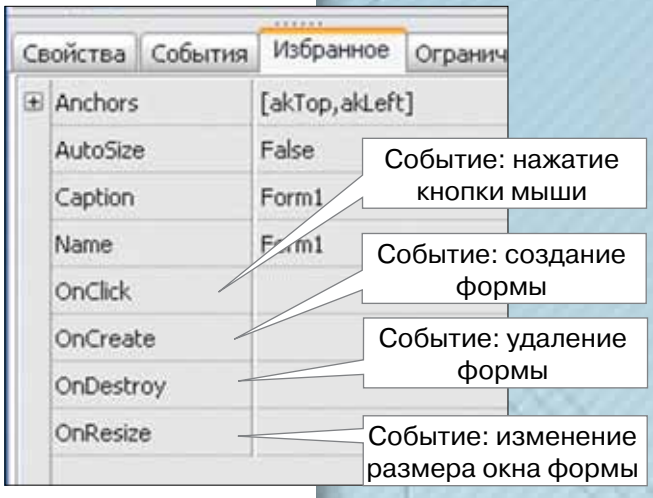


Рис. 18.3

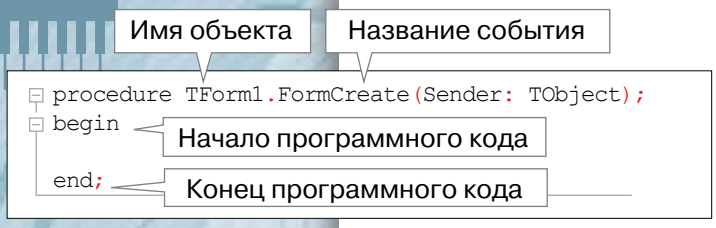


Рис. 18.4

появляется фрагмент программного кода для описания реакции на событие (рис. 18.4).

Набор команд, который используется для описания реакции на конкретное событие и записывается по правилам языка программирования, называется **процедурой**. Процедура начинается со слова `procedure`, обязательно содержит пару служебных слов, ограничивающих список команд в процедуре: `begin` — начало программного кода, `end` — конец. После служебного слова `begin` символ «;» не ставится, после `end`, так же, как после каждой строки программы, следует поставить «;».

В среде программирования *PyCharm* свойства объектов и процедуры записываются соответствующими командами в программном коде.

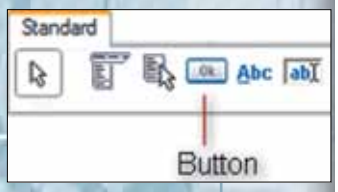


Рис. 18.5

3. Как в проекте используется элемент управления *кнопка*?

Кнопки, создаваемые в среде программирования *Lazarus* с помощью компонента `Button` (*Кнопка*) (рис. 18.5), используют для выполнения некоторого набора команд после их нажатия в режиме выполнения программы.

Кнопки имеют свойства, аналогичные свойствам других объектов (`Caption` — подпись, `Enabled` — включение, `Font` — шрифт, `Height` — высота, `Left` — отступ от края экранной формы слева, `Name` — имя, `Visible` — видимость, `Width` — ширина) и др.

С объектом *кнопка* чаще всего связывают событие `OnClick`. Чтобы перейти к редактору программного кода для ввода команд, которые будут выполняться после запуска проекта и нажатия кнопки, можно дважды щёлкнуть на кнопке в окне дизайнера формы. В окне редактора кода добавляется процедура обработки события — *нажатие кнопки* (рис. 18.6).

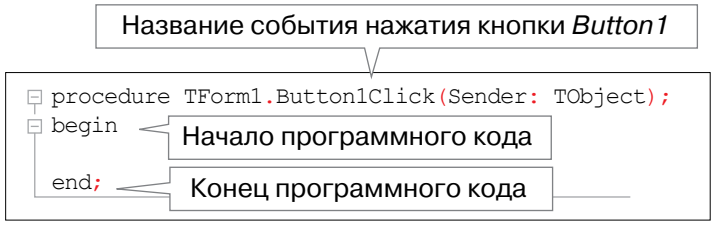


Рис. 18.6

Команды, которые необходимо выполнить в ответ на нажатие кнопки, следует записать между служебными словами `begin` и `end`.

ДЕЙСТВУЕМ

Упражнение 1. Простые и составные числа.

Задание. Измените проект *Числа*, находящийся в папке *Программирование*, так, чтобы среди натуральных чисел от 2 до 20 после нажатия кнопки *Составные числа* на экране отображались составные числа, а после нажатия кнопки *Простые числа* — простые.

1. Скопируйте папку с проектом *Числа*, находящемся в папке *Программирование*, в папку *Учебные проекты* своей структуры папок.
2. Загрузите среду программирования *Lazarus*. В меню *Проект* выберите команду *Открыть проект.*

3. Перейдите к папке *Учебные проекты*. В папке проекта *Числа* откройте файл проекта *project1.lpi.*
4. Запустите проект на выполнение. Проверьте, какие действия выполняются в программе после нажатия кнопки *Составные числа*.
5. Убедитесь, что с кнопкой *Простые числа* не связано ни одно событие в программе и при нажатии этой кнопки никакие действия не происходят.
6. Остановите выполнение программы, нажав кнопку *Завершить* .
7. Проанализируйте программный код, вызываемый событием: нажат объект `Button1` (*Кнопка1*) (рис. 18.7).
8. Перейдите в окно дизайнера форм. Дважды щёлкните на объекте `Button2` (*Кнопка2*), значение свойства `Caption` которого — *Простые числа*.
9. Перейдите в окно редактора кода. Создайте программный код аналогично соответствующему коду для объекта `Button1` (*Кнопка1*) (рис. 18.7).
10. Сохраните изменения в проекте, при необходимости отладьте его. Запустите проект на выполнение.
11. Завершите работу с проектом и средой программирования.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
      label1.Visible := False;
      label2.Visible := False;
      label3.Visible := True;
      label4.Visible := False;
      label5.Visible := True;
      label6.Visible := False;
      label7.Visible := True;
      label8.Visible := True;
      label9.Visible := True;
      label10.Visible := False;
      label11.Visible := True;
      label12.Visible := False;
      label13.Visible := True;
      label14.Visible := True;
      label15.Visible := True;
      label16.Visible := False;
      label17.Visible := True;
      label18.Visible := False;
      label19.Visible := True;
end;
```

В надписи, содержащей число 2, свойство `Visible` принимает значение `False`. Поэтому надпись **не появляется** на экранной форме

В надписи, содержащей число 4, свойство `Visible` принимает значение `True`. Поэтому надпись **появляется** на экранной форме

Название и другие свойства надписи, которые можно использовать в программном коде, отображаются при наведении на неё указателя мыши

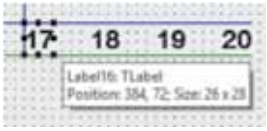


Рис. 18.7

Упражнение 2. Волшебные кнопки.

Задание. Разработайте проект, в котором при нажатии кнопки *Слева-направо* или *Справа-налево* из данных фрагментов слов «КА», «БАН» составляется слово в указанном порядке.

1. В папке *Учебные проекты* своей структуры папок создайте папку *Волшебные кнопки*.
2. Запустите среду *Lazarus*, создайте новый проект. Измените значение таких свойств объекта `Form1` (табл.18.1).
3. В окне дизайнера формы `Form1` добавьте на форму объекты, имеющие указанные значения свойств (табл. 18.2):

Таблица 18.1

Свойство	Значение свойства
`Caption`	Волшебные кнопки
`Font`	Шрифт: *Book Antiqua*, начертание: *обычный*, размер: *16*

Таблица 18.2

Объект	Свойства	Значение свойства
Label1	Caption	КА
	Font	Шрифт: *Book Antiqua*, начертание: *жирный курсив*, размер: *20*
Label2	Caption	БАН
	Font	Шрифт: *Book Antiqua*, начертание: *жирный курсив*, размер: *20*
Label3	Caption	СЛОВО
	Font	Шрифт: *Book Antiqua*, начертание: *жирный курсив*, размер: *20*
Button1	Caption	Слева-направо
	Font	Шрифт: *Book Antiqua*, начертание: *обычный*, размер: *20*
Button2	Caption	Справа-налево
	Font	Шрифт: *Book Antiqua*, начертание: *обычный*, размер: *20*

4. Дважды щёлкните на кнопке `Button1`. В окне редактора кода запишите команду, при выполнении которой свойству `Caption` надписи `Label3` будет задано значение свойства `Caption` надписи `Label1`, к которому прибавлено значение свойства `Caption` надписи `Label2`:

```
Label3.Caption := Label1.Caption+Label2.Caption;
```

5. Аналогично добавьте программный код к процедуре, которая будет обрабатывать событие — *нажатие кнопки* `Button2`. Обратите внимание, что в этом случае надпись должна формироваться в другой последовательности.
6. Сохраните проект и все входящие в него файлы. Запустите проект на выполнение. Проверьте, выполняются ли указанные в задании действия при нажатии каждой из кнопок. При наличии ошибок исправьте их.
7. Завершите работу с проектом и средой программирования.

4. Как выполнять действия с объектами с помощью методов?

Кроме процедур, составляемых программистом для обработки событий, происходящих после действий пользователя, в среде программирования *Lazarus* содержатся встроенные процедуры и функции. Некоторые из встроенных процедур являются методами объектов, с помощью которых можно выполнять действия с объектами. Для разных объектов предусмотрены разные методы, их количество и назначение зависят от объекта.

Метод — это фрагмент программного кода, встроенный в объект и предусматривающий выполнение некоторых действий с ним.

У метода есть название и набор выполняемых команд. Запись команды вызова метода состоит из имени объекта и названия метода, разделённых точкой:

```
Имя объекта.Название метода;
```

Название метода, как и название свойства, можно ввести с клавиатуры или выбрать из списка, раскрывающегося в коде программы при записи имени объекта и точки (см. рис. 18.1). В таком списке отображаются именно те методы, которые можно вызывать для выбранного объекта.

Если программа содержит несколько форм, то по умолчанию отображается только главная форма. Для отображения других форм проекта используется метод `Show` или `ShowModal`, например:

```
Form2.ShowModal;
```

Отличие этих методов заключается в том, что первый вызывают для отображения формы в обычном режиме, а второй — в так называемом **модальном** режиме: после отображения указанной формы все другие формы станут недоступными; чтобы перейти к другой форме, необходимо закрыть текущую форму.

Для объекта *форма* чаще всего используются такие методы:

- `Show` — отобразить форму в обычном режиме;
- `ShowModal` — отобразить форму в модальном режиме;
- `Hide` — сделать форму невидимой (скрыть её);
- `Close` — закрыть форму.

Некоторые методы можно вызывать для разных объектов, а другие — только для объектов определённого типа. Например, методы `Show` и `Hide` можно вызывать для формы, надписи, кнопки и других объектов, а метод `Close` — только для формы.

Если в проекте используется несколько экранных форм, то в программном коде следует задать команду `uses`, с помощью которой подключаются программные модули других форм. После команды `uses` указывают список имён файлов экранных форм, которые будут подключены. Например, если будет использована форма, содержащаяся в файле `Unit2`, то в программном коде указывают:

```
uses Unit2;
```

Вызов методов для таких «внешних» программ будет выглядеть так:

```
Имя файла.Имя объекта.Название метода;
```

ДЕЙСТВУЕМ

Упражнение 3. Справочник по физике.

Задание. Разработайте проект, в котором главная экранная форма будет выглядеть, как на рисунке18.8.

Для каждой из кнопок формы настройте обработку события нажатия кнопки так, чтобы нажатие кнопки *Завершить* закрывало окно формы. А с помощью кнопок *Теплопроводность*, *Конвекция* и *Излучение* — вызывались окна с объяснением соответствующего вида теплообмена так, что пока окно с объяснением не закрыто, к главной экранной форме перейти нельзя.

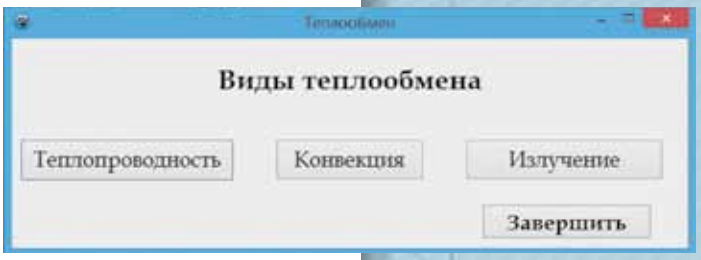


Рис. 18.8

1. В папке *Учебные проекты* своей структуры папок создайте папку *Физика*.
2. Запустите среду *Lazarus*, создайте новый проект. Измените значения свойств объекта `Form1` (табл. 18.3).
3. В окне дизайнера формы `Form1` добавьте объекты, имеющие значения свойств (табл. 18.4). Для объектов `Button1`—`Button4` задайте значения свойства `Font`: шрифт: *Book Antiqua*, начертание: *обычный*, размер: *16*
4. Добавьте к проекту форму `Form2`, размер которой соответствует размеру главной экранной формы. На экранной форме `Form2`

Таблица 18.3

Свойство	Значение свойства
`Caption`	Теплопроводность
`Font`	Шрифт: *Book Antiqua*, начертание: *обычный*, размер: *16*

Таблица 18.4

Объект	Свойство	Значение свойства
Label1	Caption	Виды теплообмена
	Font	Шрифт: *Book Antiqua*, начертание: *жирный*, размер: *20*
Button1	Caption	Теплопроводность
Button2	Caption	Конвекция
Button3	Caption	Излучение
Button4	Caption	Завершить

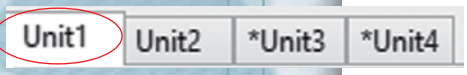


Рис. 18.9

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    Unit2.Form2.ShowModal;

end;
```

Рис. 18.10

разместите две надписи, задав для первой значение свойства `Caption`: *Теплопроводность — перенос энергии от более нагретых частей тела к менее нагретым,* для второй — *вследствие теплового движения и взаимодействия частиц.* Подберите значения свойств формы и надписей так, чтобы текст на экранной форме можно было прочитать.

5. Добавьте к проекту формы `Form3` и `Form4`, содержащие надписи: *Конвекция — перенос энергии потоками жидкости или газа* и *Излучение — перенос энергии с помощью электромагнитных волн.* Подберите значения свойств формы и надписей, как для объекта `Form2`.

6. Разместите окна дополнительных экранных форм под главной формой — друг за другом.

7. Сохраните проект и все входящие в него файлы.

8. Перейдите к вкладке редактора кода главной экранной формы `Unit1` (рис. 18.9). Введите текст в редакторе кода после команды `Implementation` (*Внедрение*):

```
uses Unit2, Unit3, Unit4;
```

9. Перейдите в окно дизайнера формы `Form1`. Дважды щёлкните на кнопке *Теплопроводность*. В окне редактора кода добавьте вызов метода `ShowModal` для формы `Form2` (рис. 18.10): показать форму, содержащуюся в файле `Unit2`, так, чтобы другие окна были недоступными.

10. Выполните аналогичные п. 9 действия для кнопок *Конвекция* и *Излучение*, по которым будут открываться соответствующие экранные формы.

11. Добавьте метод `Form1.Close;` вызываемый нажатием кнопки *Завершить*.

12. Запустите проект на выполнение. Проверьте действия, выполняемые при наступлении событий для объектов экранных форм.

13. Завершите работу с проектом и средой программирования.

5. Как в средах программирования можно использовать окна сообщений?

Для вывода данных в отдельном окне в среде программирования *Lazarus* можно воспользоваться также командой вызова окна `MessageDlg`, которая имеет такую структуру:

```
MessageDlg(сообщение, тип_окна_сообщения,
[список_кнопок], справка);
```

При этом:

- `сообщение` — текст, который будет отображён в окне сообщения;
- `тип_окна_сообщения` — внешний вид окна;
- `список_кнопок` — список данных, заданных через запятую, определяющих тип кнопки (не обязательный параметр);
- `справка` — номер окна справочной системы, выводимого на экран при нажатии клавиши *F1*. Если значение этого параметра равно нулю, то использование справки не предусмотрено.

Используют следующие типы окон сообщения: `mtInformation` (информационное), `mtWarning` (сообщение о предупреждении), `mtError` (сообщение об ошибке), `mtConfirmation` (запрос на подтверждение), `mtCustom` (обычное).

`mbYes` — *Да*
`mbNo` — *Нет*
`mbAbort` — *Прервать*
`mbRetry` — *Повторить*
`mbIgnore` — *Пропустить*
`mbHelp` — *Справка*
`mbOk` — *OK*
`mbCancel` — *Отменить*

Рис. 18.11

В окнах сообщений могут размещаться кнопки, имена которых представляются в виде списка (рис. 18.11).

В среде *PyCharm* команда создания окна сообщения записывается в программном коде и имеет вид `messagebox` (окно сообщения). Её подключают вместе с модулем графического интерфейса `tkinter` (рис. 18.12). Описание события указывают после служебного слова `def`.

Например:

```
tkinter.messagebox.showinfo("Program", "Hello Again!")
```

На языке программирования *Python* при записи следующей команды в новой строке важен отступ слева, формируемый с помощью соответствующего количества пробелов. Команды, имеющие одинаковый уровень по вертикали, независимы одна от другой и выполняются последовательно. А для команд, начинающихся правее, — каждый следующий уровень показывает, что команды этого уровня выполняются в пределах команды высшего уровня.

Например, команда

```
tkinter.messagebox.showinfo("Program", "Hello Again!")
```

в коде программы (рис. 18.12) расположена правее других команд. Это значит, что она будет выполняться в рамках обработки события, определённого командой высшего уровня `def button_click():`

```
import tkinter
import tkinter.messagebox
# обработка события нажатия кнопки
def button_click():
  tkinter.messagebox.showinfo("Program", "Hello Again!")
main = tkinter.Tk()
# создание текстовой надписи и её размещение на
главной форме
label = tkinter.Label(text="Hello World!")
label.pack()
# создание кнопки и её размещение на главной форме
button = tkinter.Button(main, text="Push Me!", command=button_
click)
button.pack()
# запуск обработки событий программы
main.mainloop()
```

Рис. 18.12

Рис. 18.13

Результат выполнения программ отображён на рисунке 18.13.

ДЕЙСТВУЕМ

Упражнение 4. Тест по биологии.

Задание. Разработайте проект *Тест*, в котором на экранной форме *Кровеносная система* размещена надпись: *Кровь выносит из клеток продукты распада, образующиеся в результате их жизнедеятельности*, — и две кнопки: *Согласен*, *Не согласен*. Если нажата кнопка, подтверждающая истинность утверждения, то появляется окно, как на рисунке 18.14, а если кнопка *Не согласен*, — как на рисунке 18.15.

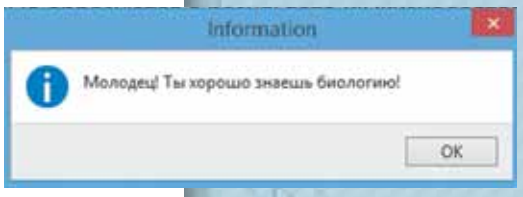


Рис. 18.14

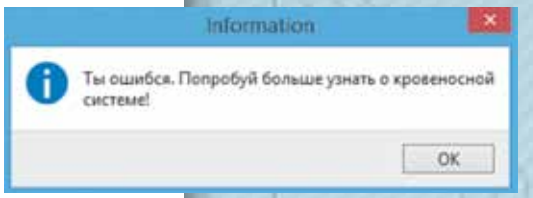


Рис. 18.15

1. В папке *Учебные проекты* своей структуры папок создайте папку *Тест*.

2. Запустите среду *Lazarus*, создайте новый проект. Измените значения свойств объекта `Form1` (табл. 18.5).
3. В окне дизайнера форм разместите объекты и задайте соответствующие значения свойствам (табл. 18.6).

Таблица 18.5

Свойство	Значение свойства
`Caption`	Кровеносная система
`Font`	Шрифт: *Book Antiqua*, начертание: *обычный*, размер: *16*

Таблица 18.6

Объект	Свойство	Значение свойства
`Label1`	`Caption`	*Кровь выносит из клеток продукты распада, образующиеся в результате их жизнедеятельности*
	`Font`	Шрифт: *Arial*, начертание: *курсив*, размер: *16*
`Button1`	`Caption`	Согласен
	`Font`	Шрифт: *Book Antiqua*, начертание: *обычный*, размер: *16*
`Button2`	`Caption`	Не согласен
	`Font`	Шрифт: *Book Antiqua*, начертание: *обычный*, размер: *16*

4. Создайте процедуру обработки события: нажата кнопка *Согласен*. В окне редактора кода введите команду создания окна сообщения:

```
MessageDlg('Молодец! Ты хорошо знаешь биологию!', mtInformation,[mbOk], 0);
```

5. Создайте процедуру обработки события: нажата кнопка *Не согласен*. В окне редактора кода введите команду создания окна сообщения:

```
MessageDlg('Ты ошибся. Попробуй больше узнать о кровеносной системе!', mtInformation,[mbOk], 0);
```

6. Запустите проект на выполнение. Проверьте, соответствуют ли условию задачи действия, связанные с объектами управления экранной формы. При наличии ошибок — исправьте их.
7. Завершите работу с проектом и средой программирования.

Обсудите вопросы, содержащиеся в файле *Тема 18* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

1. Обсудите преимущества и недостатки разных способов изменения значений свойств объектов в среде *Lazarus*.
2. Рассмотрите экранную форму некоторого проекта (рис. 18.16). Поиграйте в игру «Вопрос — ответ»: один участник задаёт вопрос, а другой на него отвечает. Если ответ неправильный или у первого участника закончились вопросы, то участники игры меняются ролями. Выигрывает тот, кто последний ответит правильно. Возможные вопросы:

Каким может быть имя указанного объекта?
Как изменить значение его свойства ...?
Какое событие может быть связано с указанным объектом?

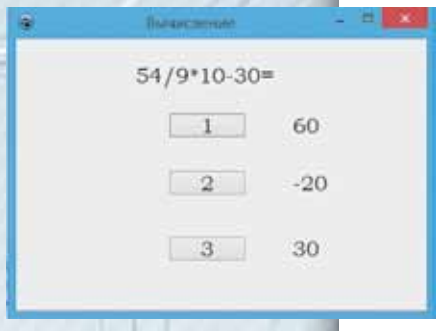


Рис. 18.16

3. Какой вид может принимать форма проекта, какие элементы управления целесообразно применить и каким образом создать проект, чтобы использовать его для обучения учеников младших классов правилам безопасного поведения в Интернете? Обсудите и разработайте дизайн окна экранной формы, опишите сценарий будущего проекта. Посоветуйтесь, какие команды можно использовать в программном коде. Распределите роли: руководитель разработки проекта и программист. Создайте спланированный проект.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Разработайте проект *Табло*, в котором на форме будут отображаться надписи с символом * (рис. 18.17, *а*) и две кнопки с цифрами 1 и 2. После нажатия кнопки с цифрой 1 или 2 на табло будут исчезать некоторые из изображённых символов и оставаться те, которые воспроизводят выбранную на кнопке цифру (рис. 18.17, *б*, *в*).
2. В среде *Lazarus* разработайте проект *Анаграммы*, в котором после нажатия кнопки *Начать* из данных трёх букв, содержащихся в надписях, на экране образуются возможные «слова». Например, из букв Д, А, Р нужно получить: ДАР, АРД, РАД, ДРА, АДР, РДА.
3. Разработайте проект *Тест* по вашему любимому учебному предмету в среде *PyCharm*. Воспользуйтесь идеей проекта теста по биологии и используйте команды, представленные на рисунке 18.12.
4. В среде *Lazarus* разработайте проект *Моя Украина*, с помощью которого пользователь может получить сведения о столице нашего государства и её географической широте и долготе; государственном строе и дате провозглашения независимости; численности населения и общей площади территории. Воспользуйтесь при необходимости справкой из Интернета. Количество форм, их дизайн и объекты спланируйте самостоятельно.

ИССЛЕДУЕМ

Упражнение 5. Обмен надписями.
Ученик 8 класса разработал проект, в котором на экранной форме отображаются две надписи. После нажатия кнопки *Обмен* содержимое надписей меняется местами. Ученик придумал два способа описания программного кода для реализации этого события. Определите возможные способы и проверьте их экспериментально.

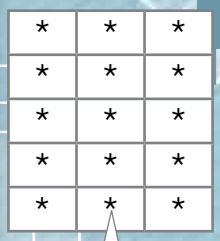


Рис. 18.17

ПОЛЕЗНЫЕ ССЫЛКИ

Материал из Википедии об использовании методов программирования:
https://ru.wikipedia.org/wiki/Метод_(программирование)

19. ПРАКТИЧЕСКАЯ РАБОТА 9

СОЗДАНИЕ ПРОГРАММЫ С КНОПКАМИ И НАДПИСЯМИ

ВСПОМНИТЕ

- Как использовать элементы управления *кнопка* и *надпись* на экранных формах;
- как изменять значение свойств объектов в программном коде;
- как использовать окна сообщений в программном коде.

СОЗДАЙТЕ

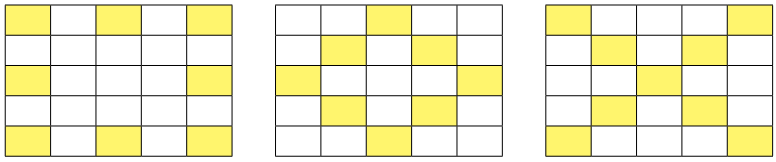
В своей структуре папок создайте папку *Практическая работа 9.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

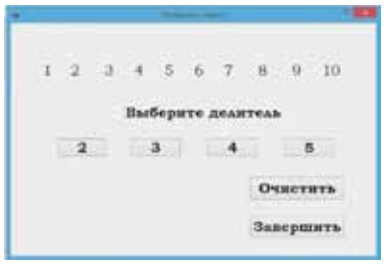
Задание 1. Мозаика (*12 баллов*)

Измените проект *Мозаика* так, чтобы некоторое слово, заданное на форме надписью *Label1*, после нажатия на кнопки *Квадрат*, *Ромб*, *Диагональ* отображалось в надписях, размещённых в позициях, которые выделены на рисунке жёлтым цветом в соответствии с формой выбранной фигуры.

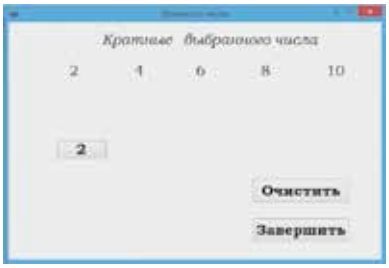


Задание 2. Делимость чисел (*20 баллов*)

Разработайте проект *Делимость чисел*, в котором из первых 10 натуральных чисел, размещённых на экранной форме, после нажатия кнопки с выбранным делителем будут отображаться только кратные 2, или 3, или 4, или 5. Кнопка *Очистить* возвращает форму к первоначальному виду, а *Завершить* — закрывает окно формы.



Первоначальное окно



Окно после нажатия кнопки с цифрой 2

Задание 3. Справочник (*12 баллов*)

В выбранной самостоятельно среде программирования разработайте проект *Справочник* с использованием окон сообщений, с помощью которого пользователь может получить сведения о применении комбинаций клавиш *Ctrl+C*, *Ctrl +V*, *Ctrl +X* при использовании буфера обмена в офисных программах. Количество форм, окон сообщений, их дизайн и объекты спланируйте самостоятельно.

РАЗДЕЛ 6

АЛГОРИТМЫ РАБОТЫ С ОБЪЕКТАМИ И ВЕЛИЧИНАМИ

20. ВЕЛИЧИНЫ, ИХ ТИПЫ И СВОЙСТВА

ВСПОМНИТЕ:

- как связаны информация, сообщение и данные;
- какие существуют типы данных;
- какие устройства использует человек для работы с данными.

ВЫ УЗНАЕТЕ:

- что такое величина и какие свойства она имеет;
- какие бывают типы величин;
- как описывают величины числового типа на языках программирования;
- как обеспечить ввод данных пользователем при выполнении программы;
- как описать выполнение операций над числовыми величинами на языке программирования;
- какие стандартные функции можно использовать в числовых выражениях.

Интересно

Понятие величины́ используется и в математике. Впервые свойства величины чётко были сформулированы Евклидом в его «Началах» (III в. до н.э.).

Интересно

При написании программ на языке *Free Pascal* в идентификаторах не различают прописные и строчные буквы. Например, имена *a1* и *A1* — воспринимаются программой как одинаковые имена одной и той же величины. А в программах, написанных на языке *Python*, наоборот, имена *a1* и *A1* отличаются и могут быть использованы для обозначения разных величин.

ИЗУЧАЕМ

1. Что такое величина и какие свойства она имеет?

Для описания объектов и процессов в материальном мире мы используем величины. С примерами величин вы сталкиваетесь ежедневно: расстояние между домом и школой, температура воздуха и т. п. С помощью величин можно указать длину отрезка, площадь земельного участка, высоту дома, скорость пешехода или автомобиля, время обращения планеты вокруг Солнца. Каждая величина характеризуется определённым значением и единицами, в которых измеряется это значение, например, скорость может быть равна 80 км/ч, расстояние — 700 м, а температура — 15 °С. Понятие величины играет важную роль в науке и отображает возможность фиксировать разные состояния некоторых объектов, в частности, количественную сторону проявлений окружающей действительности. **Величина** имеет имя и может принимать различные значения из некоторого множества допустимых значений. Тип этих значений определяет тип самой величины.

Определять значения некоторых величин можно их непосредственным измерением и по определённому алгоритму, если значения могут меняться. Так, алгоритм решения уравнения используют для определения значений неизвестных величин — корней уравнения. Компьютерная модель автомобиля, представленная математическим уравнением, позволяет определить значение величины, соответствующее расходам топлива, в зависимости от его скорости. С определением значений величин связаны также алгоритмы получения или создания текстов, различных списков и т. п. В этих случаях величины принимают значения, соответствующие фрагментам текста, элементам списков, значениям *Истина* или *Ложь* и т. п.

Для использования величин в выражениях при написании программы используют их **имена**. Обозначения имён называются также **идентификаторами**. Идентификаторы выбирают в виде некоторого конечного упорядоченного набора букв и цифр, начинающегося с буквы или символа подчёркивания _. Примерами идентификаторов величин могут быть такие последовательности символов: *A, B2C, _15, X, Y, SI, My_program, DAT_33* и т. п. Как правило, в про-

граммировании величинам присваивают имена, которые объясняют их тип и роль в программе.

Величины делятся на переменные и постоянные (константы).

Величина, имеющая одно и то же значение в любой момент времени, называется **постоянной**, или **константой**.

Константам присваиваются значения в описательной части кода программы, и в процессе выполнения программы их изменять нельзя. Для описания констант на языке программирования *Free Pascal* используют служебное слово `const` (рис. 20.1). В языке программирования *Python* константы задаются в тексте программы.

```
const
Max=1000;
Vxid='сегмент 5';
```

Рис. 20.1

Существуют константы, к значениям которых можно обращаться в программе без предварительного описания (табл. 20.1).

Таблица 20.1

Идентификатор	Значение	Описание
`True`	`True`	Истина
`False`	`False`	Ложь
`Maxint`	32767	Максимальное целое

Интересно

Служебное слово *const* — это сокращение от англ. *constant* — постоянная.

Величина, принимающая разные значения в разные моменты времени, называется **переменной**.

При выполнении программы в любой момент времени величина, как правило, имеет некоторое значение, называемое **текущим** значением. При этом переменная величина может иметь лишь одно значение или не иметь ни одного. В процессе выполнения программы величине может быть не присвоено конкретного значения. Тогда величина остаётся неопределённой.

2. Какие бывают типы величин?

Все переменные и постоянные величины относятся к определённому типу.

Числовые величины — это величины, принимающие из значения числовых множеств. Например, целая числовая величина *A* может иметь произвольные значения из множества целых чисел (..., –3, –2, –1, 0, 1, 2, 3, ...).

Величины с текстовыми значениями могут иметь символьные или строковые типы. **Символьные величины** могут принимать значения из некоторого множества символов, и каждое значение может содержать только один символ. **Строковые величины** — это величины, которые могут принимать значения из некоторого множества последовательностей символов, в частности, слов или наборов слов. Например, (‘*понедельник*', ‘*вторник*', ..., ‘*воскресенье*') — множество значений строковой величины с именем *День_недели*. **Логические величины** могут принимать только одно из двух значений: `True` (истина) или `False` (ложь).

От типа значений величин зависит множество допустимых операций. Например, нельзя выполнять арифметические операции с текстовыми величинами, операции деления и вычитания — с величинами логического типа.

Тип величины — это совокупность множества допустимых значений и операций, которые можно выполнять с этими значениями.

Тип величины определяет объём памяти, необходимый для хранения её значений, а также структуру данных.

Тип величины характеризует как постоянные, так и переменные величины (рис. 20.2).

Рис. 20.2

3. Как описывают величины числового типа на языках программирования?

Для записи арифметических выражений, аргументов математических и других функций могут использоваться числа или величины числового типа. Значения этих величин могут принимать целые или действительные числа.

Для описания числовых величин на языке программирования *Free Pascal* используют несколько служебных слов. Это зависит от диапазона их значений и соответственно размера памяти компьютера, который они могут занимать. При выполнении программ, написанных на языке *Python*, система сама определяет объём, который числовые величины могут занимать в памяти компьютера, в зависимости от введённых их значений (табл. 20.2).

Таблица 20.2

Язык программирования	Тип	Описание числовых величин на языке программирования	Возможное значение
Free Pascal	Целый	`byte`	0.. 255
		`integer`	-2147483648.. 2147483647
		`smallint`	-32768.. 32767
	Действительный	`real`	Действительное число
Python	Целый	`int`	Произвольное целое число
	Действительный	`float`	Действительное число

Действительные десятичные числа на языках программирования можно записать по правилам арифметики, при этом в такой записи целая часть от дробной отделяется десятичной точкой, а не запятой. Если десятичная точка отсутствует, число считается целым. Перед числом может записываться знак «+» или «–». Если знак отсутствует, по умолчанию число считается положительным. Например, 9.23, 0.05. Для использования очень больших или очень малых чисел их представляют в стандартном виде $a \cdot 10^n$ и записывают в виде *aEn*, где обозначение *En* заменяет умножение на 10 в степени *n*. Например, число –0,0000017 можно записать в виде –1.7E–6 (рис. 20.3).

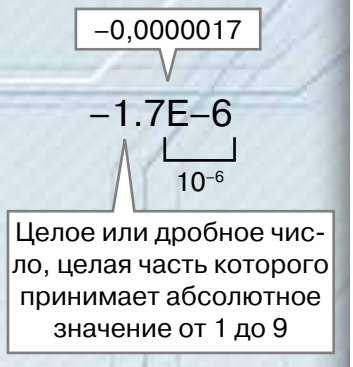


Рис. 20.3

Перед тем как использовать переменные величины в программе, описывают имя и тип каждой переменной.

В среде *Lazarus* каждая переменная должна быть обязательно описана в соответствующем месте программы. Для этого после заголовка процедуры перед

телом процедуры, ограничивающимся служебными словами `begin` и `end`, размещают раздел объявления переменных, который начинается служебным словом `var`:

```
var имя_переменной: тип_переменной;
```

Если описывается несколько переменных одного типа, то их имена записывают через запятую.

```
var переменная_1, … ,переменная_N: тип_переменных;
```

Если программа будет содержать переменные разных типов, то служебное слово `var` записывают один раз, а перечень переменных другого типа в новой строке (рис. 20.4).

```
var a: real;
    b, c: integer;
```

Переменная с именем *a* действительного типа

Переменные с именами *b* и *c* целого типа

Рис. 20.4

Язык программирования *Python* достаточно гибок и позволяет объявлять переменные в любом месте кода программы. Но корректный стиль оформления программ предусматривает описание всех переменных в одном месте программы и присваивание им начальных значений. Это удобнее для программиста, который сразу видит, переменные каких типов будут использованы. Если конкретные значения не известны или не нужны в начале работы, можно присвоить специальное значение `None` — «ничего». То есть это не ноль, не единица, не пустая строка, а просто отсутствие значения.

4. Как обеспечить ввод данных пользователем при выполнении программы?

Чтобы при выполнении программы пользователь мог ввести некоторые данные, в программе на языке *Python* можно использовать функцию `input()`. Для этого необходимо записать команду присваивания значения этой функции переменной соответствующего типа. При этом следует учитывать, что введённая величина по умолчанию будет определяться как строковая. Чтобы преобразовать её в целочисленный тип, используют функцию `int`, в действительный — `float`. Например,

- `n=int(input('Введите количество:'))` — значение переменной *n* — целое число, введённое с клавиатуры;
- `t=float(input('Введите значение нормальной температуры тела человека:'))` — значение переменной *t* — действительное число, введённое с клавиатуры;
- `a=input('Как тебя зовут?')` — значение переменной *a* — строковая величина, введённая с клавиатуры.

В среде *PyCharm* в окне выполнения программы данные вводят после соответствующего сообщения.

В среде *Lazarus* для ввода данных пользователем используется функция `InputBox()`, значение которой задают переменной соответствующего типа. Результатом использования этой функции, как и функции `input()` на языке *Python*, является значение строкового типа. В результате выполнения команды присваивания, содержащей такую функцию, на экран будет выводиться **окно ввода**, содержащее заголовок, текст подсказки и поле ввода, в которое с клавиатуры следует ввести необходимое значение. Можно также предусмотреть присваивание переменной значения по умолчанию, если пользователь не введёт

Интересно

Служебное слово *var* — это сокращение от англ. *variable* — переменная.

Интересно

Данные в компьютере хранятся в ячейках памяти, каждая из которых имеет свой адрес. При разработке компьютерных программ изначально неизвестно, в каких ячейках будут храниться те или иные данные. Поэтому при написании программ на языке программирования используют переменные, позволяющие программисту при составлении программы обращаться не к адресам ячеек памяти, а к идентификаторам.

соответствующее значение с клавиатуры. Текст заголовка окна, подсказки для ввода данных и значения по умолчанию являются аргументами функции `InputBox()`:

```
Переменная := InputBox('Заголовок', 'Подсказка', 'Значение
по умолчанию');
```

Если необходимо преобразовать значение строкового типа в числовое или наоборот, используют функции, описанные в таблице 20.3.

Таблица 20.3

Функция	Назначение функции
`StrToInt` (*значение строкового типа*)	Преобразование значения строкового типа в значение целочисленного типа
`StrToFloat` (*значение строкового типа*)	Преобразование значения строкового типа в значение действительного типа
`FloatToStr` (*значение действительного типа*)	Преобразование значения действительного типа в значение строкового типа
`IntToStr` (*значение целого типа*)	Преобразование значения целого типа в значение строкового типа

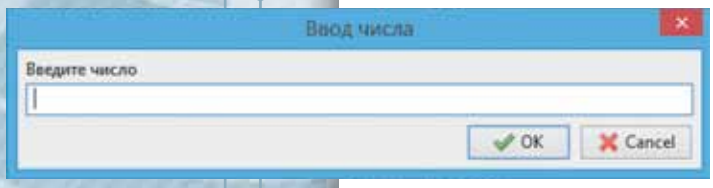


Рис. 20.5

Например, чтобы переменная *n* при вводе данных пользователем с клавиатуры получила целочисленное значение, в редакторе кода записывают команду:

```
n := StrToInt(InputBox('Ввод
числа', 'Введите число:', ''));
```

В результате выполнения такой команды будет отображаться окно ввода (рис. 20.5). Если после ввода данных в поле ввода пользователь выберет кнопку *OK*, то переменной *n* будет присвоено значение, введённое пользователем. Если будет нажата кнопка *Cancel*, то переменная *n* получит значение, указанное как значение по умолчанию.

ДЕЙСТВУЕМ

Упражнение 1. Туристическое агентство.

Задание. В среде программирования *Lazarus* разработайте для туристического агентства проект регистрационной формы для путешествия по городам Европы, в которой турист может указать следующие сведения:

- персональный код клиента;
- предполагаемый месяц путешествия;
- количество дней путешествия;
- количество лиц, путешествующих вместе с ним;
- тип питания: 0,5 — завтрак или ужин; 1,0 — трёхразовое питание; 1,5 — пятиразовое питание.

После ввода данных на экране должно появляться подтверждение введённых данных. Формой регистрации можно управлять с помощью кнопок: *Начать регистрацию*, которая вызывает окно ввода данных, *Послать форму*, которая закрывает окно главной формы.

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Форма_турагентство*.
3. Запустите среду *Lazarus*, создайте новый проект. Измените значение свойств объекта `Form1`, разместите на форме объекты и присвойте значения их свойств по образцу на рисунке 20.6.

4. Создайте процедуру обработки события: нажата кнопка *Начать регистрацию*. В окне редактора кода опишите переменные, которые будут использованы в проекте:

```
kod, month, day, number: integer;
feeding: real;
```

5. В окне редактора кода запишите команды ввода значений пользователем для всех переменных величин. Например, для переменной, принимающей значение введённого персонального кода, эта команда будет иметь вид:

```
kod := StrToInt(InputBox('Ввод данных',
'Введите персональный код:', ''));
```

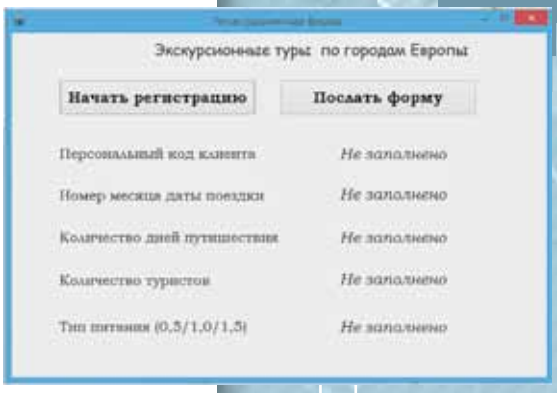


Рис. 20.6

6. Запишите команды изменения значения свойства *Caption* всех надписей, для которых начальное значение этого свойства — *Не заполнено*. Например, для надписи, соответствующей персональному коду клиента:

```
Label7.Caption := IntToStr(kod);
```

7. Создайте процедуру обработки события: нажата кнопка *Отправить форму*. В программном коде процедуры используйте метод *Закрыть форму.*
8. Запустите проект на выполнение. Проверьте, соответствуют ли действия, связанные с объектами управления экранной формы, условию задания. При наличии ошибок — исправьте их.
9. Завершите работу с проектом и средой программирования.

5. Как описать выполнение операций над числовыми величинами на языке программирования?

Вы уже знаете, как в языках программирования *Free Pascal* и *Python* используется оператор присваивания. Его также используют и для присваивания значений переменным (табл. 20.4).

Таблица 20.4

Язык программирования	Оператор присваивания	Команда присваивания	Пример
Free Pascal	:=	*<имя переменной>:=<выражение>*	*a*:=5
Python	=	*<имя переменной>=<выражение>*	*a*=5

В программе на языке *Python* присваивать значения нескольким переменным можно одной командой (рис. 20.7).

$$\underbrace{a, b, c}_{\text{имена переменных}} = \underbrace{7.5, 2.3, 8}_{\text{список значений}}$$

Рис. 20.7

Выражение, расположенное справа от символа оператора присваивания, задаёт порядок выполнения операций над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков арифметических операций, например, $a+b*sin(cos(x))$. Результат вычисления выражения присваивается переменной, имя которой записано слева. Вложенность выражения не ограничивается. Значение выражения должно быть согласовано с типом переменной, которой оно присваивается. Например, переменной действительного типа можно присвоить значение действительного или целого типа, а переменной целого — только значение целого типа.

Над числовыми величинами можно выполнять:

1) арифметические операции сложения (+), вычитания (–), умножения (*), деления (/);
2) операции целочисленной арифметики (применяются только к данным целого типа) (табл. 20.5).

Таблица 20.5

Операция		Назначение	Пример	
Free Pascal	*Python*		*Free Pascal*	*Python*
`div`	`//`	деление нацело (возвращает целую часть дроби, дробная часть отбрасывается)	`17 div 10 = 1`	`17//10 = 1`
`mod`	`%`	остаток от деления	`17 mod 10 = 7`	`17%10 = 7`

В программе на языке *Python* есть некоторые особенности использования операций сложения и вычитания:

x+=3 — увеличение значения переменной *x* на 3;

x–=2 — уменьшение значения переменной *x* на 2.

ДЕЙСТВУЕМ

Упражнение 2. Сумма цифр.

Задание. В среде программирования *Lazarus* разработайте проект *Сумматор*, в котором после нажатия кнопки *Старт* в окне ввода данных следует ввести трёхзначное целое число, после чего в окно сообщения будет выводиться сумма его цифр.

Таблица 20.6

Идентификатор	Тип	Назначение
`chislo`	`integer`	Введено трёхзначное число
`dig1`	`integer`	Первая цифра числа
`dig2`	`integer`	Вторая цифра числа
`dig3`	`integer`	Третья цифра числа
`s`	`integer`	Сумма цифр

1. Спланируйте проект. Предусмотрите, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.

2. В папке *Учебные проекты* своей структуры папок создайте папку *Сумматор*. Выполните другие шаги подготовительного этапа разработки проекта.

3. В окне дизайнера формы среды *Lazarus* добавьте объект — кнопка, присвойте значение её свойству *Caption*: *Старт.*

4. Создайте процедуру обработки события: нажата кнопка *Старт*. Опишите используемые переменные величины (табл. 20.6).

5. Запишите команду ввода значения для переменной *chislo*:

```
chislo := StrToInt(InputBox('Ввод числа',
'Введите трёхзначное число:', ''));
```

6. Проанализируйте и введите программный код выделения цифр числа:

```
dig1 := chislo div 100;
chislo := chislo - dig1 * 100;
dig2  := chislo div 10;
dig3 := chislo - dig2 * 10;
s := dig1 + dig2 + dig3;
```

7. Добавьте к программному коду команду создания окна сообщения:

```
MessageDlg(IntToStr(s), mtInformation,[mbOk], 0);
```

8. Запустите проект на выполнение. Проверьте: для введённого числа 732 вы должны получить результат 12.

9. Завершите работу с проектом и средой программирования.

6. Какие стандартные функции можно использовать в числовых выражениях?

В выражениях, записываемых в среде программирования для выполнения вычислений, можно использовать стандартные функции, некоторые из них вы уже использовали на уроках математики или будете изучать позднее (табл. 20.7).

Таблица 20.7

Функция	Тип аргумента	Тип результата	Результат
`abs(x)`	Целый / действительный	Целый / действительный	Модуль числа
`sin(x)`	Действительный	Действительный	Синус числа
`cos(x)`	Действительный	Действительный	Косинус числа
`pi`	Без аргумента	Действительный	Значение числа π
`sqr(x)`	Действительный	Действительный	Квадрат числа
`sqrt(x)`	Действительный	Действительный	Квадратный корень
`int(x)`	Действительный	Действительный	Целая часть числа
`frac(x)`	Действительный	Действительный	Дробная часть числа
`round(x)`	Действительный	Целый	Округление числа
`trunc(x)`	Действительный	Целый	Отбрасывание дробной части числа
`random(n)`	Целый	Целый	Случайное число от 0 до *n*

Чтобы использовать указанные функции в программе на языке *Python* в среде *PyCharm*, в программном коде обращаются к библиотеке `math`:

```
from math import cos, pi
y = cos(pi)
```

ДЕЙСТВУЕМ

Упражнение 3. Дорожки.

Задание. В городе все пешеходные дорожки построены по перпендикулярным линиям. Но этот путь не кратчайший. Разработайте проект в среде *PyCharm*, с помощью которого можно будет предоставить мэрии этого города новую модель и расчёт длины новых коротких дорожек.

1. Спланируйте проект. Определите исходные данные и результат. Постройте информационную модель задания. Обратите внимание, что ею может быть графическая модель (рис. 20.8).
2. Запустите среду программирования *PyCharm*.
3. Создайте новый файл программы на языке *Python* с именем *Дорожки* в папке *Учебные проекты* своей структуры папок.
4. В окне редактора кода введите команду подключения библиотеки математических функций:

```
from math import sqrt
```

5. Запишите в окне редактора кода команды для ввода значений для переменных *a*, *b*:

```
a = int(input('a = '))
b = int(input('b = '))
```

6. Запишите выражение для определения значения переменной *c* и команду для вывода результата на экран согласно правилам языка *Python*:

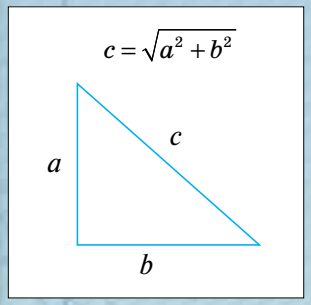


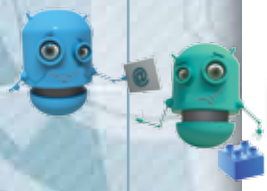
Рис. 20.8

```
c = sqrt(a*a+b*b)
print(c)
```

7. Запустите проект на выполнение. Проверьте, равен ли *5.0* результат для значений переменных $a = 3$, $b = 4$.
8. Завершите работу с проектом и средой программирования.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 20* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.



РАБОТАЕМ В ПАРАХ

1. Предложите друг другу примеры числовых величин, значения которых содержатся в диапазонах, описанных в таблице 20.2. Один называет описание типа величины на языке программирования, другой — пример.
2. Поиграйте в игру «Значение величины — тип величины». Один участник называет множество значений некоторой величины, а другой — соответствующий тип величины. Игра продолжается до первой ошибки, после чего участники меняются ролями.
3. Предложите друг другу выражения для вычисления значений переменной-результата, в которых исходными данными являются целые числа, а результатом — действительное число. Запишите программный код объявления переменных и вычисления значения переменной-результата: один из участников — на языке программирования *Free Pascal*, другой — на языке *Python*. Обменяйтесь записями и проверьте друг друга.
4. Обсудите, как изменить проект *Дорожки*, созданный в среде *PyCharm*, чтобы реализовать его в среде *Lazarus*. Результаты обсуждения используйте при создании проекта.
5. Обсудите, как изменить проект *Сумматор*, созданный в среде *Lazarus*, чтобы реализовать его в среде *PyCharm*. Результаты обсуждения используйте при создании проекта.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Запишите на языках программирования *Free Pascal* и *Python* выражения для вычисления количества мест K в самолёте при изменении количества рядов, если:
   а) в самолёте несколько рядов пассажирских сидений. В каждом ряду R находится четыре места;
   б) в самолёте два места в кабине экипажа (пилот и второй пилот) и несколько рядов пассажирских сидений. В каждом ряду R находится четыре места;
   в) в самолёте два места в кабине экипажа (пилот и второй пилот), несколько рядов R_1 1-го класса и несколько рядов R_2 2-го класса пассажирских сидений. В каждом ряду 1-го класса находится четыре места, в каждом ряду 2-го класса находится пять мест.

   Создайте соответствующие выражения в учебной онлайновой среде *Блокли* по адресу:
   https://blockly-demo.appspot.com/static/demos/plane/index.html?lang=ru

Используйте блоки для записи арифметических операций сложения и умножения, записи количества рядов и числовых значений (рис. 20.9). Изменяя положение ползунка для изменения количества рядов, определите количество мест в самолёте, если в самолёте будет 8, 12, 17 рядов для заданий *а*) и *б*) (рис. 20.10) и 3, 4 или 7 рядов первого класса — для задания *в*).

Рис. 20.10

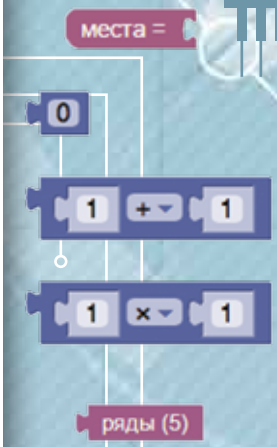


Рис. 20.9

2. Разработайте проект *Семейный депозит*, с помощью которого можно определить, какую сумму получит семья в конце года, если в его начале откроет в банке депозитный счёт в размере S тыс. грн под 18 % годовых. Среду программирования выберите самостоятельно. Проверьте выполнение проекта для $S = 10\,000$.

3. В средах программирования *Lazarus* и *PyCharm* разработайте проект *Электричество*, с помощью которого можно определить сопротивление электрической цепи R, если в ней проводники с сопротивлениями $R1$, $R2$, $R3$, $R4$ соединены:

1) последовательно: $R = R1 + R2 + R3 + R4$;

2) параллельно: $R = \frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3} + \frac{1}{R4}$.

Сравните программные коды определения значения переменной R. Проверьте выполнение проекта для $R1 = 2$, $R2 = 3$, $R3 = 4$, $R4 = 5$ Ом.

4. Для швейной компании разработайте проект с графическим интерфейсом *Остаток*, с помощью которого пользователь планового отдела будет определять площадь ткани, оставшейся после вырезания круга радиуса R из заготовки в форме квадрата со стороной a. Проверьте выполнение проекта для $a = 4$ м, $R = 2$ м.

5. Даны переменные x, y действительного типа. Разработайте проект для вычисления значения переменной z для $x = 1$, $y = -5$:

1) $z = \frac{|x| + |y|}{1 + |xy|}$;

2) $z = 1 + |y - x| + \frac{(y - x)^2}{2} + \frac{|y - x|^3}{3}$;

3) $z = \frac{2y}{\cos\left(x - \frac{\pi}{6}\right)}$;

4) $z = \frac{x}{1 + \frac{x^2}{3 + (2x)^2}}$.

6. Для ведения домашнего бизнеса спланируйте и разработайте проект *Домашняя бухгалтерия*, в котором можно ввести доходы всех членов семьи за месяц и обязательные выплаты из бюджета на коммунальные платежи, транспорт, связь, Интернет, другие необходимые расходы и рассчитать остаток семейного бюджета на следующий месяц. Дизайн и элементы управления подберите самостоятельно. Для проверки выполнения проекта введите данные о бюджете своей семьи.

21. ПРАКТИЧЕСКАЯ РАБОТА 10

СОСТАВЛЕНИЕ И ВЫПОЛНЕНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ РАБОТЫ С ВЕЛИЧИНАМИ В СРЕДЕ ПРОГРАММИРОВАНИЯ

ВСПОМНИТЕ

- Как описывают переменные величины на языке программирования;
- как вводить значения величин в программе;
- какие операции можно выполнять с числовыми величинами.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 10.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Парк аттракционов (*11 баллов*)

В парке аттракционов построили новую горку длиной c, которая наклонена к основанию b под углом α градусов. Разработайте проект *Аттракцион*, с помощью которого можно определить высоту горки a и длину основания b, округлённые до целого числа. Используйте формулу для перевода величины из градусов в радианы:

$$\text{радианы} = \text{градусы} \cdot \alpha/180.$$

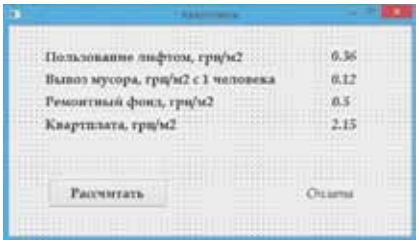
Задание 2. Длительность рейса (*11 баллов*)

Разработайте проект *Длительность рейса* для вычисления длительности рейса в часах и минутах для некоторого автотранспортного предприятия, если известно, что маршрут состоит из трёх отрезков — $s1$, $s2$, $s3$ км, которые транспорт преодолевает со скоростью $v1$, $v2$, $v3$ км/ч. Между участками маршрута есть остановки по t мин.

Задание 3. Квартплата (*18 баллов*)

В объединении совладельцев многоквартирного дома осуществляются следующие выплаты: 0,36 грн/м^2 — за пользование лифтом, 0,12 грн/м^2 — за вывоз мусора с человека, 0,5 грн/м^2 — формирование ремонтного фонда, 2,15 грн/м^2 — квартплата.

В среде программирования *Lazarus* разработайте проект *Квартплата* для расчёта оплаты в соответствии с показателями квитанции за квартиру площадью S м^2 за месяц, в которой проживает n человек. Предусмотрите ввод значений площади и количества лиц в окнах ввода, а отображение размера оплаты — на форме проекта.



Задание 4. Преобразование текста (*10 баллов*)

Для преобразования текста в среде программирования *PyCharm* разработан проект *Кодирование*, находящийся в папке *Программирование*. Реализуйте проект для тестовых значений, выбранных в соответствии с командами программы. Установите, какой алгоритм преобразования текста использован в программе.

22. ТЕКСТОВЫЕ ВЕЛИЧИНЫ И ОПЕРАЦИИ С НИМИ

ВСПОМНИТЕ:

- какие объекты содержит текстовый документ;
- как редактировать текст;
- как работать с документами в текстовых процессорах.

ВЫ УЗНАЕТЕ:

- как вводить и описывать текстовые величины на языках программирования;
- какие операции выполняют с текстовыми величинами;
- какие функции используют для работы с текстовыми величинами;
- какими могут быть ошибки при создании и выполнении программ;
- как отладить программу в средах программирования.

ИЗУЧАЕМ

1. Как вводить и описывать текстовые величины на языках программирования?

Для работы с текстовыми данными на языке программирования используют величины символьного и строкового типов.

Значением величины символьного типа может быть один символ — буква, цифра или знак. Набор символов образует строку.

Присваивать значения текстовым величинам можно разными способами:

- при написании программного кода с помощью оператора присваивания;
- при выполнении программы, содержащей команды или окна ввода;
- с помощью элемента управления *текстовое поле*.

Текстовое поле создаётся на экранной форме проекта в среде *Lazarus* с помощью компонента *Edit* (*Поле ввода*) (рис. 22.1).

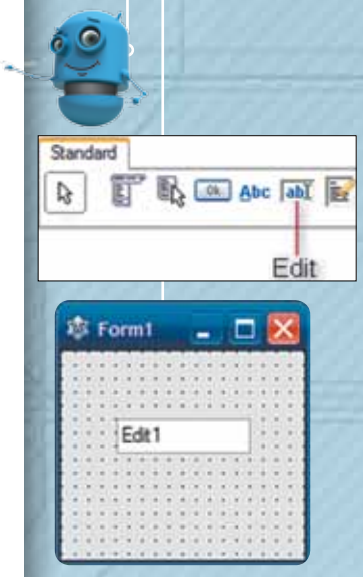


Рис. 22.1

Текст, записанный в поле ввода, определяется значением свойства `Text` (*Текст*), и его можно редактировать после запуска проекта на выполнение.

Кроме свойств, которые имеют другие элементы управления, например `Label` (*Надпись*), для текстового поля можно определить дополнительные (табл. 22.1).

Таблица 22.1

Свойство	Описание
`ParentFont`	Наследование значений параметров шрифта формы, на которой расположен компонент. Если это свойство имеет значение `True`, то при изменении свойства `Font` формы автоматически изменяется значение свойства `Font` компонента *текстовое поле*
`Enabled`	Ограничение возможности изменить текст в текстовом поле. Если это свойство имеет значение `False`, то текст в текстовом поле редактировать нельзя

В программах, созданных на языке программирования *Python*, после подключения модуля оконного графического интерфейса пользователя также можно использовать текстовое поле для ввода данных. Для этого предназначена функция `tkinter.Entry` (рис. 22.2).

```
edit = tkinter.Entry(main, textvariable = str_var)
edit.pack()
```

Переменная

Объект *текстовое поле*

Текстовая величина

Значение текстового поля

Размещение объекта *текстовое поле* на экранной форме

Объект *графическая программа*

Рис. 22.2

Текстовые величины, как и числовые, описывают в программе в разделе описания переменных, указывая соответствующий тип. На языках программирования для описания текстовых величин используются служебные слова (табл. 22.2).

Таблица 22.2

Тип величины	Описание на языке программирования	
	Free Pascal	*Python*
Символьный	`char`	`str`
Строковый	`string`	

Например, описание переменных на языке программирования *Free Pascal* представлено на рисунке 22.3.

```
var str: string[20];
t: string;
s: char;
```

Переменная строкового типа, содержащая до 20 символов

Переменная строкового типа, содержащая до 255 символов

Переменная символьного типа

Рис. 22.3

```
print('''День
первый''')
```

```
День
первый
```

Рис. 22.4

В программном коде на языке программирования *Free Pascal* текстовое значение записывают в одинарных кавычках ('). Например:

```
s:='Q'; s1:='data';
```

В программном коде на языке программирования *Python* можно использовать одинарные и двойные кавычки. Если для обозначения строки применены двойные кавычки, внутри можно свободно использовать одинарные, а двойные кавычки интерпретатор будет считать концом строки. И наоборот, если строка обозначена одинарными кавычками, внутри могут содержаться двойные. Кроме того, для обозначения строк можно использовать три пары одинарных или двойных кавычек. В таком случае всё, что содержится между ними, будет отображаться так, как записано. Такой способ присваивания значений текстовым величинам удобно использовать для последующего структурированного вывода данных (рис. 22.4).

Строки могут содержать наборы символов, начинающиеся с наклонной чёрточки \ — они называются **escape-последовательностями**. При выводе строки такие символы обрабатываются специальным образом и могут полностью или частично не отображаться на экране. Наиболее часто используются следующие:

- \n — переход на новую строку;
- \t — вставка табуляции;
- \" — двойные кавычки (при необходимости вставить двойные кавычки в тексте, заключённом в двойные кавычки);
- \' — одинарные кавычки (при необходимости вставить апостроф в строку, заключённую в одинарные кавычки).

У всех символов в строке есть свой порядковый номер. По этому номеру можно определить значение символа, указав номер в квадратных скобках после имени перемен-

Интересно

Понятие **escape-последовательности** происходит от англоязычного аналога *escape sequences* (от англ. *escape* — избегать), используется для записи в текстовых значениях специальных символов, которые запрещают вывод на экран самой последовательности символов.

Таблица 22.3

Язык программирования	Переменная	Символ
Free Pascal	`z := 'школа'`	`z[4]` — л
Python	`z = 'школа'`	`z[4]` — а

ной. В языке программирования *Free Pascal* нумерация символов начинается с единицы, а в *Python* — с нуля (табл. 22.3).

ДЕЙСТВУЕМ

Упражнение 1. Поздравительная открытка.

Задание. В среде программирования *Lazarus* разработайте проект *Поздравительная открытка* по образцу (рис. 22.5), в котором в соответствующие текстовые поля на экранной форме пользователь вводит имя адресата приветствия, событие и имя автора поздравления.

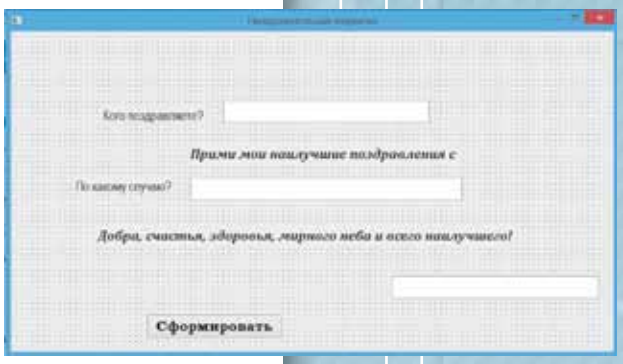


Рис. 22.5

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В среде программирования *Lazarus* разработайте проект *Поздравительная открытка* по образцу.
3. Запустите среду *Lazarus*, создайте новый проект. Измените значения свойств объекта `Form1`, разместите на форме объекты и измените значения их свойств, чтобы после запуска проекта на выполнение можно было получить приветствие, например, представленное на рисунке 22.6.
4. Создайте процедуру обработки события: нажата кнопка *Сформировать*. В окне редактора кода опишите переменные, которые будут использованы в проекте: `person` (адресат), `event` (событие), `nik` (имя автора). Укажите их тип — `string` (строковый). Проанализируйте фрагмент программного кода (рис. 22.7).

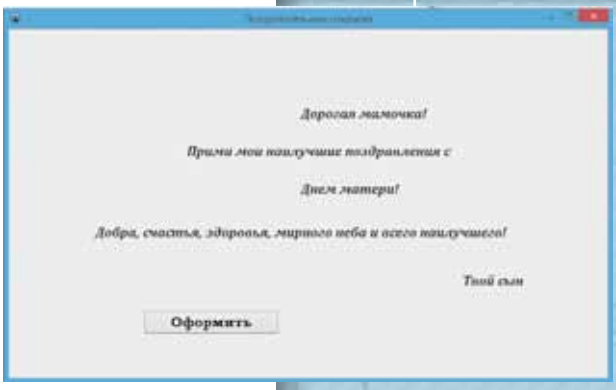


Рис. 22.6

Переменной *person* присвоить значение, вводимое в текстовом поле

```
person := Edit1.Text;
Label5.Caption := person;
Label5.Visible := True;
Edit1.Visible := False;
Label1.Visible := False;
```

Содержимое текстового поля перенести в соответствующую надпись

Надпись с заданным текстом становится видимой

«Лишние» элементы управления сделать невидимыми

Рис. 22.7

5. В окне редактора кода запишите команды, с помощью которых значениям свойства `Caption` надписей 5–7 на форме присвоен текст, введённый в текстовые поля экранной формы. При этом текстовые поля и надписи с вопросами станут невидимыми.
6. Запустите проект на выполнение. Проверьте, соответствуют ли действия, связанные с объектами управления экранной формы, условию задания. При наличии ошибок — исправьте их.
7. Завершите работу с проектом и средой программирования.

Упражнение 2. Напоминание.

Задание. В среде программирования *PyCharm* разработайте проект, в котором пользователь будет вводить день недели, нажимать кнопку *Принять* и получать в окне с заголовком *Внимание*! такое сообщение:

Сегодня — <день недели, который был введён>.

1. Запустите среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Напоминание* в папке *Учебные проекты* своей структуры папок.
3. В окне редактора кода запишите команды, содержащиеся в файле *Код_Напоминание* в папке *Программирование*.
4. Измените значения свойств объектов, используемых в программном коде, так, чтобы они реализовали задание.

```
import tkinter
import tkinter.messagebox
main = tkinter.Tk()
# создание объекта для получения значения из текстового поля
str_var = tkinter.StringVar()
# обработка события нажатия кнопки
def button_click():
    tkinter.messagebox.showinfo("Program", "Hello" + str_var.get())
# создание текстовой надписи и её размещение на главной форме
label = tkinter.Label(text="Enter Your Name:")
label.pack()
# создание текстового поля и его размещение на главной форме
edit = tkinter.Entry(main, textvariable = str_var)
edit.pack()
# создание кнопки и размещение объекта на главной форме
button = tkinter.Button(main, text="Push Me!",
command=button_click)
button.pack()
# запуск обработки событий программы
main.mainloop()
```

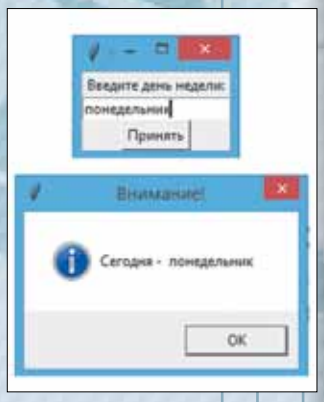


Рис. 22.8

5. Запустите проект на выполнение. Проверьте, соответствует ли рисунку 22.8 результат для введённого дня недели *понедельник*.
6. Завершите работу с проектом и средой программирования.

Таблица 22.4

Язык программирования	Пример набора команд	Результат — значение переменной *R*

Free Pascal	`F := 'алго';`
	`L := 'ритм';`
	`R := F+L;`
		`'алгоритм'`

Python	`F = 'алго'`
	`L = 'ритм'`
	`R = F+L;`
		`'алгоритм'`

2. Какие операции выполняют с текстовыми величинами?

С текстовыми величинами выполняют операцию склеивания — соединение нескольких строковых величин, которая обозначается символом «+» (табл. 22.4).

Текстовые величины являются неизменяемыми, это значит, что нельзя изменить часть строки, не создав новой.

В программе на языке программирования *Python* при работе с текстовыми величинами можно использовать операцию среза, с помощью которой копируется последовательность или её часть. Например, для переменной `s = 'Hello world'` операции среза описаны в таблице 22.5.

Таблица 22.5

Описание	Пример команды	Результат — значение переменной *s1*
Срез от начала строки до символа с номером *n*	`s1 = s[:6] + 'Python!'`	`'Hello Python!'`
Срез от символа с номером *n* строки включительно до символа с номером *m*	`s1 = s[3:5]`	`'lo'`
Срез от символа с номером *n* строки включительно до конца строки	`s1 = s[6:]`	`'world'`

Чтобы записать символы в строке в обратном порядке, используют операцию `s3 = s[::-1]`.

Тогда переменная `s3` будет иметь значение `'dlrow olleH'`.

ДЕЙСТВУЕМ

Упражнение 3. Слова.

Задание. Составьте программу на языке *Python*, с помощью которой из введённого слова *информатика* будут образовываться слова *форма*, *романтика*.

1. Запустите среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Слова* в папке *Учебные проекты* своей структуры папок.
3. Допустим, в программе будут использованы переменные: *s* — значением которой будет введённое с клавиатуры слово *информатика*; *s1* — которая должна получить значение *форма*; *s2* — которая должна получить значение *романтика*.
4. Определите номер позиции каждой буквы в слове.

0	1	2	3	4	5	6	7	8	9	10
и	н	ф	о	р	м	а	т	и	к	а

5. В окне редактора кода запишите команды ввода переменной *s* и определения значений переменных *s1*, *s2*:

```
s = input('Введите слово')
s1 = s[2:7]
s2 = s[4] + s[3] + s[5:7] + s[1] + s[7:]
```

6. Запишите команды вывода полученных значений.
7. Запустите проект на выполнение. Введите в окне выполнения программы значение переменной *s*: *информатика*. Проверьте полученные результаты. При наличии ошибок — исправьте их.
8. Завершите работу со средой.

Упражнение 4. Дата рождения.

Задание. Разработайте проект в среде *Lazarus*, в котором в текстовые поля, размещённые в верхней части формы, пользователь вводит день, месяц и год своего рождения и после нажатия кнопки *Пуск* в соответствующей текстовой надписи формируется дата рождения в формате *день.месяц.год*.

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Дата_рождения*.
3. Запустите среду *Lazarus*, создайте новый проект. Измените значение свойства `Caption` объекта `Form1`, присвоив ему значение *Дата рождения*.

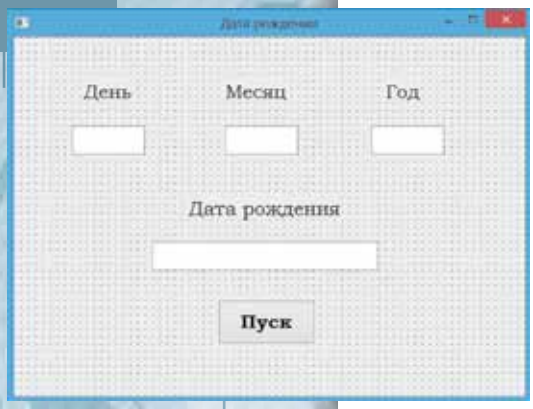


Рис. 22.9

Разместите на форме объекты по образцу и задайте произвольные значения их свойствам. Свойству `Enabled` текстового поля для вывода результата задайте значение `False` (рис. 22.9).

4. Создайте процедуру обработки события: нажата кнопка *Пуск*. В окне редактора кода опишите переменные, которые будут использованы в проекте:

```
var d, m, y, rez: string;
```

5. В окне редактора кода запишите команды присваивания значений переменным *d*, *m*, *y* по образцу:

```
d := edit1.text;
```

6. Запишите выражение для получения значения переменной *rez*:

```
rez := d + '.' + m + '.' + y;
```

7. Задайте значение полученной переменной свойства `Text` текстового поля для отображения полной даты рождения.

8. Запустите проект на выполнение. Введите данные в текстовые поля, месяц рождения запишите числом. Нажмите кнопку *Пуск* и проверьте правильность работы программы. При наличии ошибок — исправьте их.

9. Завершите работу с проектом и средой программирования.

3. Какие функции используют для работы с текстовыми величинами?

Вы уже использовали функции для преобразования текстовой величины в величину числового типа и наоборот.

В программах на языке программирования *Free Pascal* используются и другие функции для работы с текстовыми величинами (табл. 22.6).

Таблица 22.6

Описание функции на языке программирования	Тип аргумента	Тип результата	Назначение
`length(S)`	*S* — текстовый	Целый (byte)	Определение количества символов в строке *S*
`copy(S,n,m)`	S — текстовый, *n*, *m* — целый	Текстовый	Копирование *n* символов строки *S*, начиная с позиции *m*
`delete(S,n,m)`	*S* — текстовый, *n*, *m* — целый	Текстовый	Удаление *n* символов строки *S*, начиная с позиции *m*
`insert(S,S1, m)`	*S*, *S1* — текстовый, *m* — целый	Текстовый	Вставка строки *S1* в строку *S*, начиная с позиции *m*
`pos(S1,S2)`	*S1*, *S2* — текстовый	Целый	Номер позиции, с которой строка *S2* входит в строку *S1*

Интересно

UTF-8 (от англ. *Unicode Transformation Format* — формат преобразования Юникода) — кодировка, при которой текст, состоящий только из символов, код которых меньше 128, при записи в UTF-8 преобразуется в обычный текст ASCII.

При работе с текстовыми величинами, содержащими символы украинского и русского алфавитов, эти функции работают некорректно. Чтобы этого избежать, в раздел `uses` добавляют модуль *LCLProc*:

```
uses LCLProc;
```

Тогда перед именем каждой функции, представленной в таблице 22.6, добавляют *UTF8*. Например, при использовании латинских букв используется функция `length('s')`, а при использовании символов кириллицы — `UTF8length('ф')`.

В языке программирования *Python* для работы с текстовыми величинами используются функции, описанные в таблице 22.7.

Таблица 22.7

Описание функции на языке программирования	Назначение
`len(S)`	Определяет количество символов в строке *S*
`s.upper()`	Изменяет регистр всех символов строки на верхний
`s.lower()`	Изменяет регистр всех символов строки на нижний
`s.replace(s_old, s_new)`	Заменяет все вхождения фрагмента *s_old* в строке на *s_new*
`s.replace(s_old, s_new, count)`	Заменяет первые *count* вхождений фрагмента *s_old* в строке на *s_new*
`s.find(s1)`	Возвращает позицию вхождения (индекс первого символа) фрагмента *s1* в строке, или –1, если фрагмент не найден
`s.find(s1,start_pos)`	Возвращает позицию вхождения (индекс первого символа) фрагмента *s1* в строке, начиная с позиции *start_pos*, или –1, если фрагмент не найден

В языках программирования *Free Pascal* и *Python* есть одинаковые функции для работы с текстовыми величинами (табл. 22.8).

Таблица 22.8

Описание функции на языке программирования	Тип аргумента	Тип результата	Назначение
`chr(x)`	Целый	Символьный	Определяет символ с кодом *x*
`ord(c)`	Символьный	Целый	Определяет код символа *c*

Упражнение 5. Инициалы.

Задание. Разработайте в среде *Lazarus* проект, после запуска которого в текстовые поля экранной формы пользователь будет вводить фамилию, имя и отчество. После нажатия кнопки *Пуск* в соответствующем текстовом поле будут выведены фамилия и инициалы (рис. 22.10).

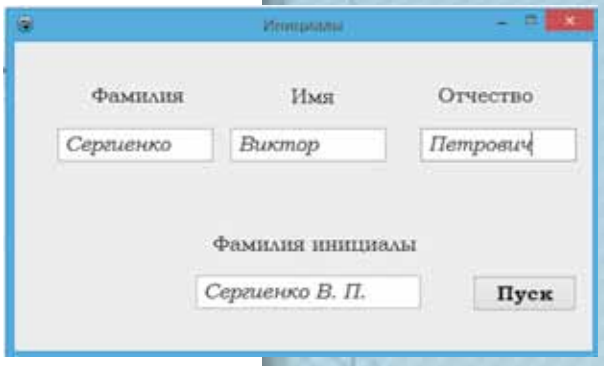


Рис. 22.10

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Инициалы*.
3. Запустите среду *Lazarus*, создайте новый проект. Измените значение свойства `Caption` объекта `Form1`, задав значение *Инициалы*. Разместите на форме объекты по образцу и задайте произвольные значения их свойствам. Свойству `Enabled` текстового поля для отображения фамилии и инициалов задайте значение `False`, чтобы пользователь не мог изменить содержимое этого поля с клавиатуры.
4. Создайте процедуру обработки события: нажата кнопка *Пуск*. В окне редактора кода опишите переменные, которые будут использованы в проекте:

```
var last_name, first_name, sname, full_name: string;
```

5. В окне редактора кода запишите команды присваивания значений переменным

```
last_name, first_name, sname.
```

6. Запишите команду определения значения переменной величины *full_name:*

```
full_name := last_name + ' ' + UTF8copy(first_name, 1, 1) + '.' + UTF8copy(sname, 1, 1) + '.';
```

7. Запишите команду для вывода полученного значения переменной в соответствующее текстовое поле.
8. Запустите проект на выполнение. Введите данные в текстовые поля. Нажмите кнопку *Пуск* и проверьте правильность работы программы. При наличии ошибок — исправьте их.
9. Завершите работу с проектом и средой программирования.

4. Какими могут быть ошибки при создании и выполнении программ?

Созданный в среде программирования проект может не выполняться вообще, или результат его выполнения не будет соответствовать ожидаемому результату. Это происходит, если при составлении алгоритма решения задачи либо написании кода программы были допущены ошибки. Различают три группы ошибок:

- синтаксические;
- ошибки выполнения;
- логические.

Синтаксические ошибки можно обнаружить как в процессе написания программного кода, так и после запуска проекта на выполнение. Если некоторая команда в программном коде написана программистом не по правилам, принятым в языке программирования, то она может отображаться другим цветом (рис. 22.11).

Команда записана правильно

```
Edit3.Text := Edit1.Text;

Edit4.Text := текст;
```

В команде значение текстовой переменной не заключено в апострофы

Рис. 22.11

После запуска проекта на выполнение, если в программном коде такая ошибка не была исправлена, в среде *Lazarus* в окне сообщения о ходе компиляции проекта отображается номер строки программного кода и позиция объекта в строке, в которой допущена ошибка, а также описание ошибки (рис. 22.12).

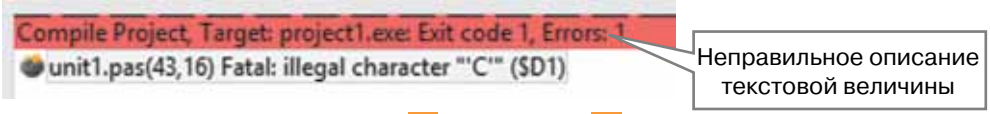


Рис. 22.12

В окне редактора кода строка, на которой «остановился» процесс компиляции, будет указана (рис. 22.13).

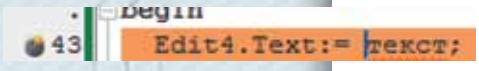


Рис. 22.13

Строка, в которой допущена ошибка, может быть указана не точно. Например, в сообщении об ошибке в строке под номером 43 в окне сообщения может быть указана строка с номером 44 (рис. 22.14).

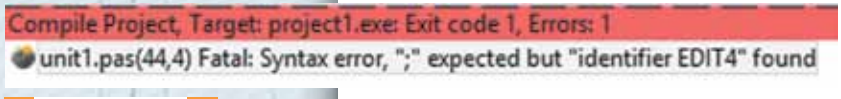


Рис. 22.14

В среде *PyCharm* при вводе команды отображается окно сообщения с подсказкой о правильном синтаксисе (рис. 22.15).

Но если программист всё-таки допустил ошибку, то в коде проекта команда, записанная не по правилам синтаксиса языка, будет выделена красной волнистой линией 

. После запуска проекта на выполнение в области выполнения программы будут отображены не только место расположения ошибки в программном коде, но и сама команда, в которой допущена ошибка (рис. 22.16).

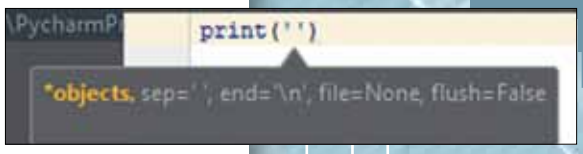


Рис. 22.15

```
File ''D:/oshibky.py'', line 1
    print('слово)
                  ^
SyntaxError: EOL while scanning string literal
```

Отсутствует символ завершения текстовой величины

Рис. 22.16

Типичными являются ошибки несоответствия типов описанных величин и значений, которые им присваиваются в процессе выполнения программы. Например, если текстовой величине `Edit1.Text` присваивается числовое значение (рис. 22.17).

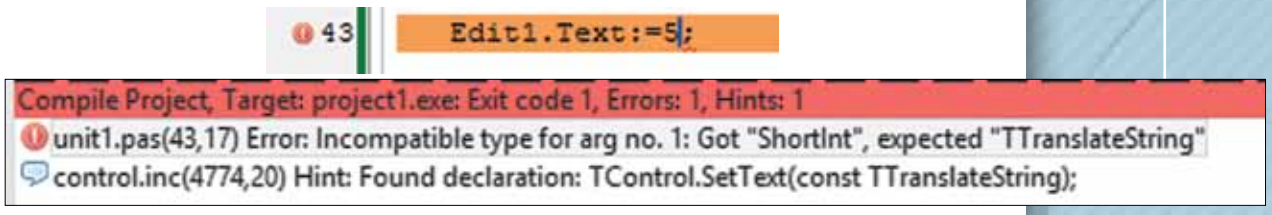


Рис. 22.17

К ошибкам выполнения относятся ошибки, связанные с неправильными числовыми вычислениями, ошибки вычисления значений величин по формулам. Их распознают только при выполнении программы. Например, если в программе на языке *Python* использовать программный код:

```
f = 5
c = 0
r = f/c
```

то после запуска программы на выполнение в окне выполнения получим сообщение:

```
r = f/c
ZeroDivisionError^ division by zero
```

В среде *Lazarus* такие ошибки называются исключениями, они сопровождаются выводом на экран сообщения об ошибке (рис. 22.18).

Логические ошибки — это ошибки используемого в программе алгоритма. Результат, полученный в ходе выполнения программы, не совпадает с ожидаемым результатом. Такие ошибки нельзя обнаружить средствами программной среды.

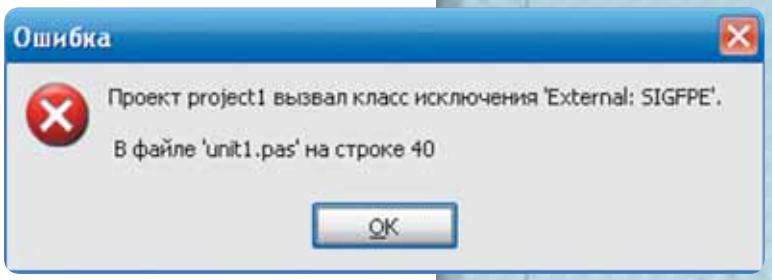


Рис. 22.18

5. Как отладить программу в средах программирования?

Исправить синтаксические ошибки и ошибки выполнения программы можно после их выявления. Затем необходимо снова запустить проект на выполнение. Для того чтобы проверить наличие логических ошибок, нужно запустить программу с тестовым (исходным) набором данных и проверить, совпадает ли ожидаемый результат «ручного» подсчёта и программного.

Процесс поиска логических ошибок в тексте программы с использованием тестовых наборов исходных данных называется **тестированием программы**.

Помогают в поиске логических ошибок комментарии, которые можно включать в текст программы. **Комментарии** — это текст, который не выполняется после запуска программы, а размещается для объяснения структуры программного кода.

На языке программирования *Python* комментарии начинаются с символа #, а на языке программирования *Free Pascal* — с символа //, если комментарий не занимает больше строки, {} или (* *) — если комментарий занимает несколько строк.

Если текст программы большой, то для поиска логических ошибок иногда удобно тестировать программу не полностью, а частями. Фрагменты программы, которые временно не нужно использовать в программном коде, помечают как многострочные комментарии. Тогда после запуска проекта на выполнение закомментированный фрагмент программного кода не будет выполняться.

Интересно

Тестовый набор данных — это значения переменных величин, используемых в программном коде, для которых известен результат выполнения программы. Тестовые наборы данных в своей работе используют тестировщики, которые вместе с программистами принимают участие в разработке программного обеспечения.

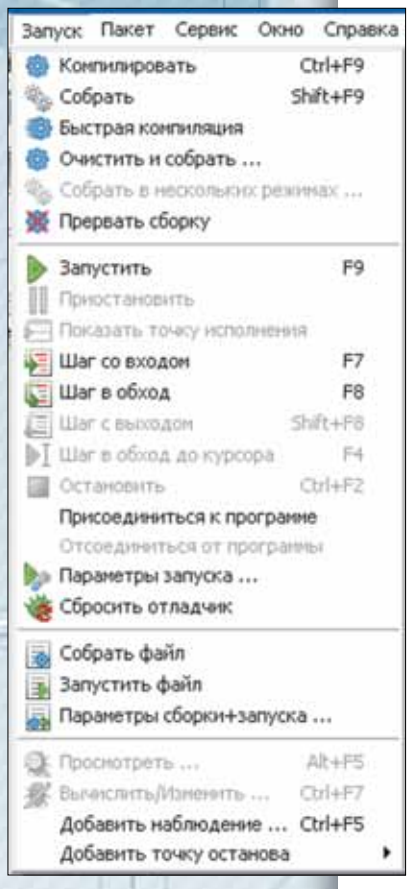


Рис. 22.19

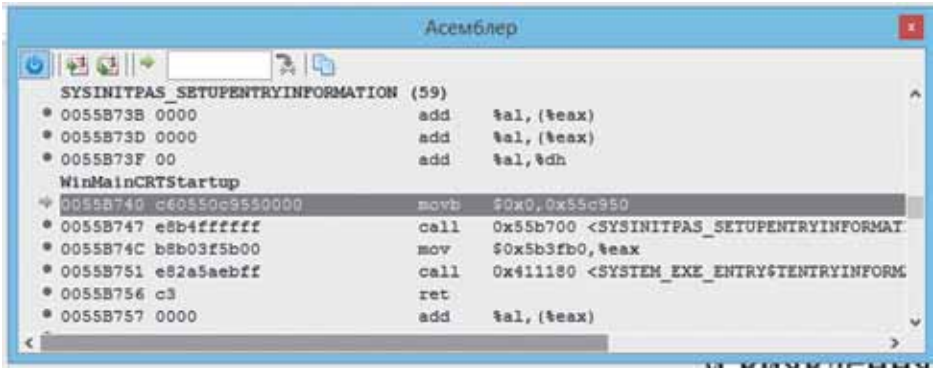


Рис. 22.20

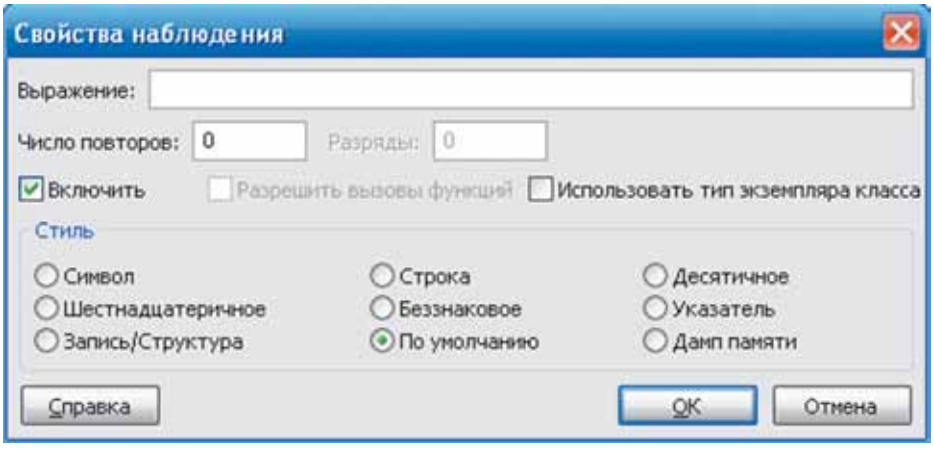


Рис. 22.21

Ещё одним способом выявления в программе логических ошибок является её пошаговое выполнение, или **трассировка**. Шаг выполнения программы — это строка программы. Для выполнения одного шага программы в среде *Lazarus* нажимают клавишу *F7* или в меню *Выполнить* выбирают команду *Шаг со входом* (рис. 22.19).

После первого нажатия клавиши *F7* происходит компиляция проекта, и если проект не содержит синтаксических ошибок, то начинается его выполнение, при этом появляется окно программы в машинных кодах (рис. 22.20).

С каждым следующим нажатием клавиши *F7* будут выполняться команды следующих строк: открывается окно программы, в которое можно ввести данные, переменным величинам присваиваются введённые значения, выполняются вычисления и т. д., в конце — выводится результат. Чтобы при выполнении программы следить за изменением значений переменных, настраивают параметры окна *Свойства наблюдения*, вызов которого осуществляется с помощью команды *Добавить наблюдение* меню *Выполнить* или нажатием клавиш *Ctrl+F5* (рис. 22. 21).

В поле *Выражение* записывают идентификаторы переменных, за которыми необходимо наблюдать в ходе пошагового выполнения программы. Значение введённых переменных будет отображаться в окне *Список наблюдений*, появляющемся после нажатия кнопки *OK* (рис. 22.22).

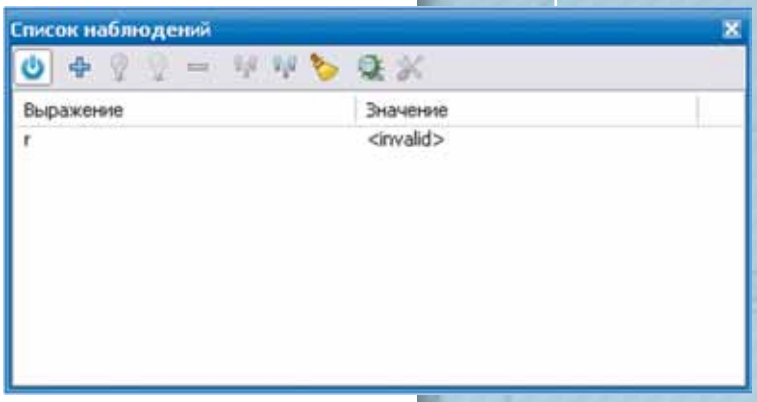


Рис. 22.22

Сравнивая значения указанной переменной с ожидаемыми, можно определить, в какой момент выполнения программы это значение не соответствует предположениям. Это может быть причиной логической ошибки.

В среде *PyCharm* отладка программы происходит аналогично. В меню *Run* (*Выполнить*) или на панели инструментов выбирают команду *Debug* (*Отладить*)  . В нижней части окна среды отображается окно отладки (рис. 22.23).

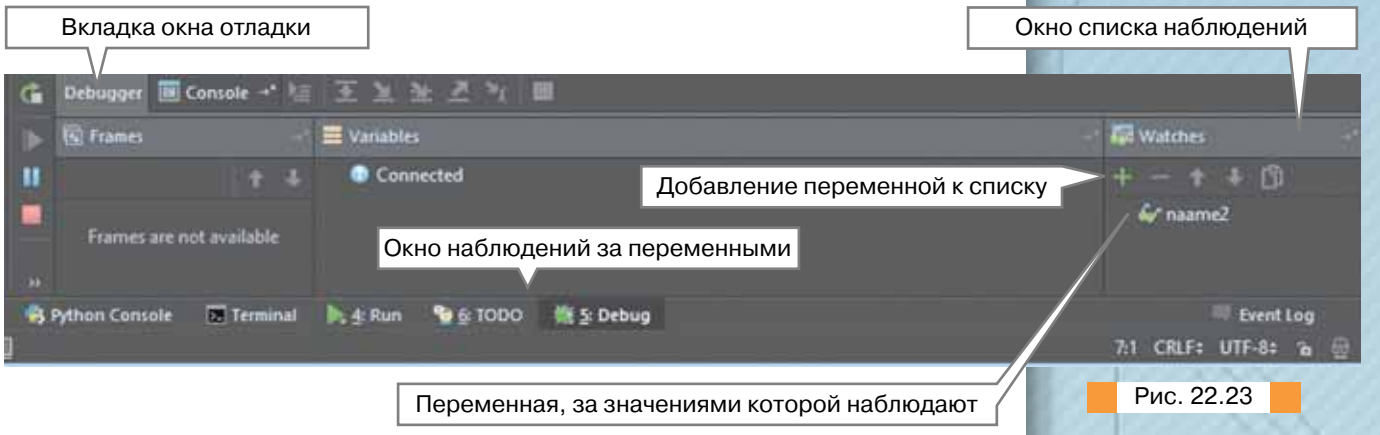


Рис. 22.23

В области редактора кода отображается пошаговый процесс выполнения программы и значения введённых или вычисленных переменных, переход к следующему шагу происходит при нажатии клавиши *F7* (рис. 22.24).

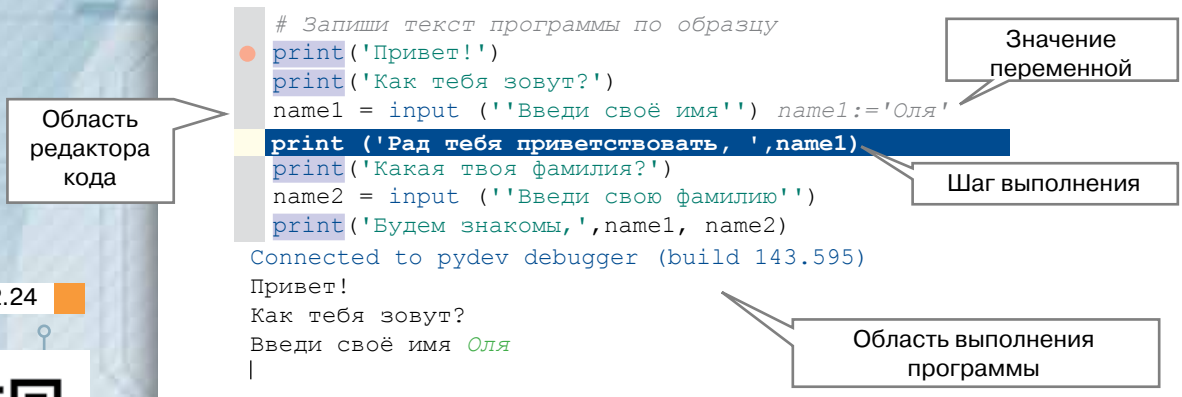


Рис. 22.24

Команды, с которыми работают при отладке программы, можно вызвать в меню *Run* (*Запуск*) или нажатием соответствующей комбинации клавиш.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 21* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Обсудите, как на языках программирования *Free Pascal* и *Python* записать данное слово в обратном порядке. Составьте каждый соответствующие программы в выбранной среде программирования и сравните их.
2. В программе, написанной на языке программирования *Free Pascal* или *Python,* используя значение текстовой переменной *s1*, получили *s2* (табл. 22.9). Составьте соответствующий программный код. Проверьте, одинаковые ли средства получения значения переменной *s2* из переменной *s1* на выбранном языке программирования использовал каждый из вас. Сделайте выводы.

Таблица 22.9

Значение переменной *s1*	**Значение переменной *s2***
Слово — не поймаешь, вылетит — не воробей	Слово — не воробей, вылетит — не поймаешь
Мирнебездобрыхлюдей	Мир не без добрых людей
Век учись — век живи	Век живи — век учись

3. Обсудите, как на языках программирования *Free Pascal* и *Python* можно вывести текст заявления о зачислении в кружок технического творчества в принятом для такого документа виде, в котором данные о заявителе вносятся в процессе выполнения программы. Определите общее и отличия.
4. Обсудите общее в процессах отладки программы в среде программирования и редактирования текста в среде текстового процессора. Можно ли считать редактирование и отладку синонимами?
5. Обсудите и, используя средства текстового процессора, создайте сравнительную таблицу процесса отладки программы в проектах, реализованных в средах *Lazarus* и *PyCharm*.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Составьте программный код, который можно использовать в средах *Lazasus* и *PyCharm*, чтобы из слова *s1* получить слово *s2* (табл. 22.10).

Таблица 22.10

№	Слово *s1*	Слово *s2*	№	Слово *s1*	Слово *s2*
1	рекомендация	оценка	3	алгоритм	игра
2	университет	турист	4	спортсмен	метро

2. В самостоятельно выбранной среде программирования разработайте проект *Шифровальщик*, в котором в текстовое поле вводят слово из пяти букв, а после нажатия кнопки *Старт* в окне сообщения получают результат по правилам, описанным в таблице 22.11.

Таблица 22.11

№	Результат	Исходное значение	Полученное значение
1	Все буквы написаны дважды	школа	шшккооллаа
2	Изменён порядок букв в парах	книга	нкгиа
3	После каждой буквы добавлена последняя буква слова	голубь	гьоьльуьбььь
4	Каждая буква заменена на соответствующий код в кодовой таблице	тетрадь	10791086109610801090
5	Каждая буква заменена на следующую в кодовой таблице	лист	мйту

3. Разработайте проект *Разрядные единицы*, в котором в текстовое поле вводят четырёхзначное целое число, нажимают кнопку *Разложить* и в текстовом поле, защищённом от изменений, получают запись числа в виде суммы разрядных единиц. Например, для введённого числа *5843* получают *5*1000+8*100+4*10+3*1.*

4. Разработайте проект *Калькулятор*, в котором в текстовые поля вводят числа, а после нажатия кнопки с обозначением математического действия в защищённом от изменений текстовом поле получают правильный результат (рис. 22.25). Задайте тестовые значения для проверки работы программы самостоятельно. Отладьте программу, используя средства среды *Lazarus*.

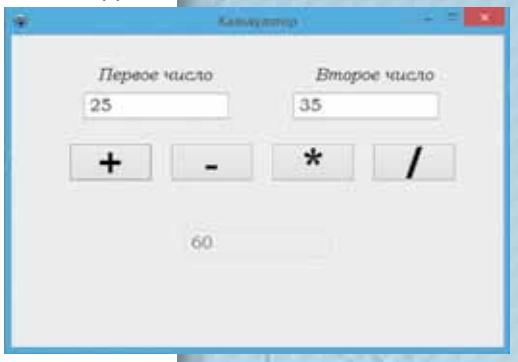


Рис. 22.25

5. Разработайте проект *Дата рождения* в среде *PyCharm*, в котором в текстовые поля пользователь вводит день, месяц и год своего рождения, а после нажатия кнопки *Пуск* в соответствующей текстовой надписи формируется дата рождения, записанная в формате дд/мм/гг. Чем будет отличаться этот проект от проекта, созданного в среде *Lazarus*?

23. ПРАКТИЧЕСКАЯ РАБОТА 11

ОТЛАДКА ГОТОВОЙ ПРОГРАММЫ

ВСПОМНИТЕ

- Как соотносятся типы данных и элементы для ввода значений величин;
- какие операции можно выполнить над числовыми и текстовыми величинами;
- как отлаживать и выполнять программы в среде программирования;
- как анализировать результаты выполнения программ.

СОЗДАЙТЕ

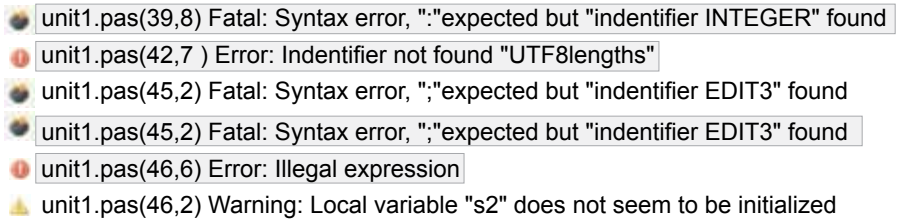
В своей структуре папок создайте папку *Практическая работа 11.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Преобразование текста (*11 баллов*)

После запуска проекта *Преобразование текста*, реализованного в среде программирования *Lazarus*, в окне сообщения пользователи получают сообщения о синтаксических ошибках:



Найдите ошибки в программном коде и исправьте их. Запустите проект на выполнение и проверьте наличие логических ошибок в программном коде.

Задание 2. Выражения (*10 баллов*)

Для вычисления значений выражений в среде программирования *PyCharm* разработан проект *Выражения*. Реализуйте проект для тестовых значений, выбранных самостоятельно, и сравните результат выполнения программы с результатом, полученным с помощью использования инженерного режима стандартной программы *Калькулятор.*

Задание 3. Таблица значений (*10 баллов*)

В программном коде проекта *Таблица значений*, реализованного в среде программирования *Lazarus* и находящегося в папке *Программирование*, нет синтаксических ошибок. Подберите тестовые значения для выявления ошибок выполнения. При необходимости внесите изменения в программный код проекта.

24. РАБОТА С ВЕЛИЧИНАМИ ЛОГИЧЕСКОГО ТИПА. КОМАНДА ВЕТВЛЕНИЯ

ВСПОМНИТЕ:

- что такое высказывания и как их классифицируют;
- правила использования разветвляющихся алгоритмов;
- описание разветвляющихся алгоритмов в полной и сокращённой формах;
- описание условия в разветвляющихся алгоритмах в среде *Скретч*.

ВЫ УЗНАЕТЕ:

- как сравнивают значения величин в программах;
- какие операции выполняют с логическими величинами;
- как описать алгоритмическую структуру ветвления на языках программирования;
- как с помощью элементов управления задать логическое значение величины;
- для чего на форме используют элемент управления *раскрывающийся список*.

ИЗУЧАЕМ

1. Как сравнивают значения величин в программах?

Вы уже умеете использовать высказывания для записи условий. Простые высказывания на языках программирования можно записать в виде логических выражений с использованием операций сравнения (табл. 24.1).

Таблица 24.1

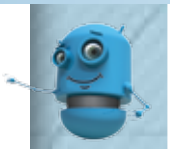
Операция	Язык программирования	
	Free Pascal	*Python*
Больше	>	>
Меньше	<	<
Не больше	<=	<=
Не меньше	>=	>=
Равно	=	==
Не равно	<>	!=

Результатом выполнения операции сравнения значений двух величин является величина **логического типа**, которая может принимать одно из двух значений: `True` или `False`. Для описания логических величин на языках программирования *Free Pascal* и *Python* используют служебные слова (табл. 24.2).

Сравнение значений величин разных типов имеет определённые особенности. Сравнение числовых величин происходит по правилам математики. Сравнение текстовых величин на языке *Free Pascal*, в результате которого получено логическое значение `True`, продемонстрировано на рисунке 24.1.

```
'Алгоритм' < 'алгоритм' – код прописной буквы А меньше кода буквы а;
'алгоритм' > 'алго' – длина первой величины больше длины второй;
'алгоритм' < > 'alhoritm'
'алг' = 'алг'.
```

Рис. 24.1



Интересно

Логический тип получил своё название в честь английского математика и логика середины XIX в., одного из основателей математической логики Джорджа Буля.

Таблица 24.2

Free Pascal	*Python*
`boolean`	`bool`

Для логических величин $a =$ `True`, $b =$ `False` результатом выполнения операции $a > b$ будет значение `True`, т. к. истинное значение `True` интерпретируется как 1, а ложное `False` — 0. Очевидно, что $1 > 0$.

2. Какие операции выполняют с логическими величинами?

Кроме операций сравнения, с логическими величинами на всех языках программирования для записи составных высказываний выполняют логические операции:

- `not` (*не*) — отрицание;
- `and` (*и*) — конъюнкция;
- `or` (*или*) — дизъюнкция (нестрогая);
- `xor` (*исключающие или*) — дизъюнкция (строгая).

Результаты выполнения этих операций с переменными *A* и *B* логического типа, принимающими значения `True` (1) и `False` (0), представлены в таблице истинности (табл. 24.3).

Таблица 24.3

A	B	not A	A and B	A or B	A xor B
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

Из таблицы истинности видно, что с помощью логической операции `not` изменяют значение логической величины на противоположное. Результат операции `and`, совпадающий с результатом умножения значений 1 и 0, является истинным лишь при условии, что обе величины *A* и *B* принимают значение `True` — истина. Результат операции `or` будет ложным только при условии, что обе величины *A* и *B* принимают значение `False`, и истинным — во всех других случаях. Например, если $A = 5$, а $B = 7$, то значением логического выражения $(A < B)$ `and` $(B = 7)$ является `True`, т. к. логические выражения $5 < 7$ и $B = 7$ принимают значение `True`, и поэтому по таблице истинности результатом операции `and` является значение `True`.

Чтобы определить значение составного логического выражения, содержащего несколько логических операций, используют таблицу истинности и учитывают приоритет выполнения операций: в первую очередь выполняется операция `not`, далее — `and`, в последнюю очередь — `or` и `xor`. Как и для числовых выражений, для изменения порядка выполнения логических операций используются скобки.

В языке программирования *Python* логические значения `True` и `False` можно преобразовать в значения других типов. Например, при преобразовании их в строковые величины получим слова `'True'` и `'False'` соответственно. Для преобразования логических значений в значения числовых типов используют функции преобразования `int`, `float`. В этом случае `True` соответствует единице, а `False` — нулю как для целых, так и для действительных чисел (рис. 24.2).

В обратном порядке преобразование отличается, но его правила запомнить просто: любые «непустые» значения конвертируются в `True`, любые «нулевые» — в `False`. Для непосредственного присваивания значения логического типа используется встроенная функция `bool` (рис. 24.3).

Интересно

Операцию `or` также называют логическим сложением, а `and` — логическим умножением. Названия «конъюнкция» и «дизъюнкция» происходят от англ. *conjunction* — объединение и *disjunction* — разъединение. В математике и логике для записи логических операций приняты специальные обозначения. Операцию `and` обозначают $\wedge$, а операцию `or` — $\vee$.

```
int(True) = 1
int(False) = 0
float(True) = 1.0
float(False) = 0.0
str(True) = 'True'
str(False) = 'False'
```

Рис. 24.2

```
bool(None) = False
bool(1) = True
bool(-1.1) = True
```

Рис. 24.3

ДЕЙСТВУЕМ

Упражнение 1. Таблица истинности для логического выражения.

Задание. Определите, какое значение может принимать логическое выражение

(*not A*) *or* (*B and A*)

в зависимости от значений, принимаемых логическими переменными *A* и *B*.

1. Вычислим количество возможных наборов значений логических переменных по формуле $N = 2^n$, где N — количество наборов значений, n — количество переменных. В нашем случае $N = 2^2 = 4$. Это число определяет количество строк в таблице истинности.
2. Определим количество и порядок логических операций в выражении.

```
      1     3      2
(not A) or (A and B)
```

Количество логических операций и логических переменных определяет количество столбцов в таблице истинности. Для нашего случая: 3 + 2 = 5.

3. Построим таблицу истинности (табл. 24.4). Для удобства записи примем `True` = *1*, `False` = *0*. Для определения значений каждой логической операции используем таблицу истинности 24.3.

Таблица 24.4

A	***B***	***1***	***2***	***3***
		not A	***A* and *B***	***1* or *2***
1	1	0	1	1
1	0	0	0	0
0	1	1	0	1
0	0	1	0	1

4. Сделаем вывод: логическое выражение принимает значение `False` только тогда, когда переменная *A* принимает значение `True`, а переменная *B* — значение `False`. Во всех других случаях логическое выражение будет иметь значение `True`.

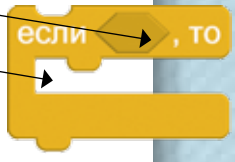
3. Как описать алгоритмическую структуру ветвления на языках программирования?

Простые и составные логические выражения, значениями которых являются `True` или `False`, используются в описании алгоритмической структуры ветвления. С её помощью исполнитель алгоритма может выбрать один из сценариев последующих действий в зависимости от выполнения определённого условия.

Для описания алгоритмической структуры ветвления на языке программирования, как и в среде *Скретч*, используют **оператор ветвления в сокращённой форме** (табл. 24.5) и **оператор ветвления в полной форме** (табл. 24.6).

Таблица 24.5

Язык программирования	Описание
Free Pascal	`if <логическое выражение>` `then <команда>;`
Python	`if <логическое выражение>:` `  блок команд`



```
begin
    <команда 1>;
    <команда 2>;
    …
    <команда n>;
end;
```

Рис. 24.4

Команда или блок команд в операторе ветвления в сокращённой форме будет выполняться только тогда, когда логическое выражение принимает значение `True`.

В операторе ветвления в сокращённой форме на языке программирования *Free Pascal* после служебного слова `then` можно записать только одну команду или блок команд, ограниченных операторными скобками `begin` … `end` (рис. 24.4).

На языке программирования *Python* в операторе ветвления в сокращённой форме после логического выражения записывают символ «:» и следующую строку начинают с отступа на 4 символа. Все команды, имеющие такой же отступ в программном коде, входят в блок команд, выполняемых в случае, если условие истинно.

Для записи оператора ветвления в полной форме дополнительно используют служебное слово `else` (табл. 24.6).

Таблица 24.6

Язык программирования	Описание

Free Pascal	`if <логическое выражение>`
	`then <команда 1>`
	`else <команда 2>;`

Python	`if <логическое выражение>:`
	`  <блок команд 1>`
	`else:`
	`  <блок команд 2>`

Если необходимо учитывать значения нескольких логических выражений, то используют вложенные ветвления (табл. 24.7).

Таблица 24.7

Язык программирования	Описание

Free Pascal	`if <логическое выражение 1> then <команда 1>`
	`else if <логическое выражение 2> then <команда 2>`
	…
	`else if <логическое выражение n> then <команда n>`
	`else <команда иначе>;`

Python	`if <логическое выражение1>:`
	`  <блок команд 1>`
	`elif <логическое выражение2>:`
	`  <блок команд 2>`
	`...`
	`elif <логическое выражение n>:`
	`  <блок команд n>`
	`else:`
	`  <блок команд иначе>`

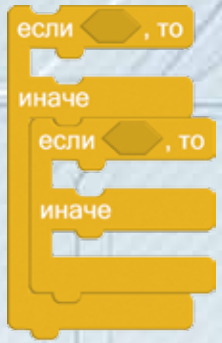


Рис. 24.5

Если в описанных вложенных ветвлениях значение логического выражения 1 — `True`, то выполняется `<команда 1>` или `<блок команд 1>`. Если значение логического выражения 1 — `False` и значение логического выражения 2 —

`True`, то выполняется `<команда 2>` или `<блок команд 2>` и т. д. В противном случае выполняется команда `<иначе>` или `<блок команд иначе>` (рис. 24.5).

ДЕЙСТВУЕМ

Упражнение 2. Агрегатное состояние воды.

Задание. Разработайте проект в среде *Lazarus*, в котором по введённому значению температуры воды будет определяться её агрегатное состояние.

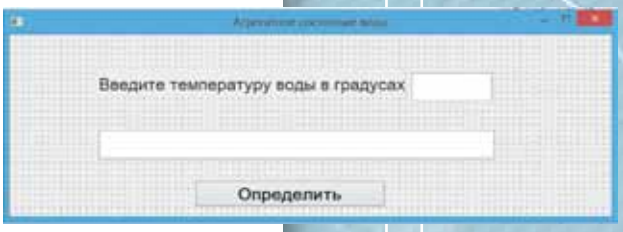


Рис. 24.6

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Состояние_воды*.
3. Откройте среду *Lazarus*, создайте новый проект и сохраните его составляющие в папку *Состояние_воды*. Измените значения свойств объекта `Form1`, разместите на форме необходимые объекты и задайте значения их свойствам, чтобы получить форму, как на рисунке 24.6.
   Обратите внимание, что для всех объектов используется значение свойства шрифта — *Arial Unicode MS*, размер — 16. Начертание определите по рисунку. Задайте значение `False` свойству `Enabled` текстового поля для вывода значения.
4. Создайте процедуру обработки события нажатия кнопки *Определить*. В окне редактора кода введите программный код (рис. 24.7). Обратите внимание на структуру оператора ветвления.

```
procedure TForm1.Button1Click(Sender: TObject);
var t:integer;
begin
t := StrToint(Edit1.Text);
if t<0 then Edit2.Text := 'Вода находится в твёрдом состоянии'
   else if t>100 then Edit2.Text := 'Вода находится в газообразном состоянии'
           else Edit2.Text := 'Вода находится в жидком состоянии';
end;
```

Рис. 24.7

5. Запустите проект на выполнение. Введите значение температуры, например, 25. Проверьте полученный результат. Определите, какое условие соответствует полученному значению.
6. Запустите проект на выполнение ещё раз для значения –10,5. Объясните, почему проект не выполняется. Внесите изменения в проект так, чтобы пользователь мог задавать температуру воды действительным числом. Обратите внимание, что действительное число, вводимое в текстовое поле, записывается с десятичной запятой, а в программном коде используется число, записанное с десятичной точкой.
7. Сохраните изменения в проекте. Завершите работу с проектом и средой программирования.

4. Как с помощью элементов управления задать логическое значение величины?

Для реализации ветвления в проекте можно использовать элементы управления: флажок `CheckBox` или переключатель `RadioButton` (рис. 24.8).

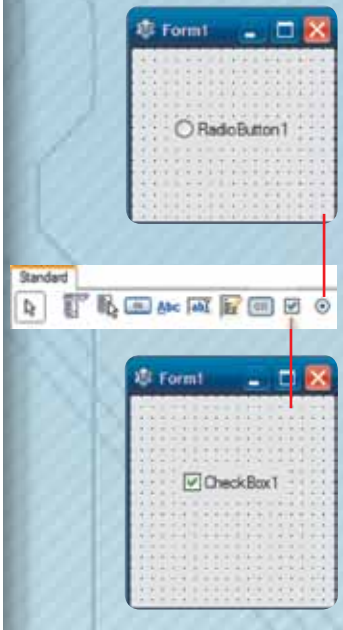


Рис. 24.8

Элементы управления CheckBox и RadioButton могут находиться в одном из двух состояний: включён и выключен. Поэтому с их помощью в программный код можно передать логическое значение True — включён, или False — выключен.

На одной форме можно использовать несколько включённых флажков CheckBox, но только один включённый переключатель RadioButton.

Кроме стандартных свойств, например, таких как Caption, Font, которые вы уже использовали для других элементов управления, компонент CheckBox обладает специальными (табл. 24.8).

Таблица 24.8

Свойство	Описание
Checked	Изменение состояния флажка: если значение этого свойства *True*, то флажок включён; если значение свойства *False*, то флажок выключен (нет отметки ✓). Значение изменяется автоматически, хотя его также можно изменить в программном коде
State	Состояние флажка. Позволяет задавать состояние включения флажка по одному из параметров: включён cbChecked, выключен cbUnChecked, промежуточное состояние (серый) cbGrayed

С элементом управления CheckBox связаны события OnChange и OnClick.

Событие OnClick возникает всякий раз, когда пользователь включает или выключает флажок на форме после запуска программы на выполнение. Событие OnChange также возникает, когда пользователь включает или выключает флажок. В отличие от события OnClick, происходящим только при щелчке на флажке, событие OnChange происходит в любом случае, когда изменяется состояние флажка, — если пользователь мышью включил-выключил флажок или такое изменение предусмотрено в программном коде, выполняемом после нажатия некоторой кнопки, с помощью команды:

```
ChB1.Checked := not ChB1.Checked;
```

Тогда при нажатии такой кнопки состояние флажка изменится на противоположное — состоится OnChange, а событие OnClick — нет, ведь мышью на флажке не щёлкали.

Чтобы добавить к программному коду процедуру обработки одного из этих событий, можно дважды щёлкнуть на нём в таблице окна *Инспектор объектов*.

Значение свойства Checked элемента управления RadioButton, определяющее состояние переключателя, не может быть изменено в программном коде, в отличие от аналогичного значения свойства элемента управления CheckBox.

Кроме компонентов CheckBox и RadioButton, на форму можно добавить компоненты CheckGroup или RadioGroup (рис. 24.9). Их используют в случае, если необходимо разместить несколько групп флажков или переключателей. Если на форме использовано несколько групп переключателей, в пределах каждой из групп можно включить по одному.

С переключателями и флажками в группе выполняют те же действия, что и с отдельными элементами управления на форме.

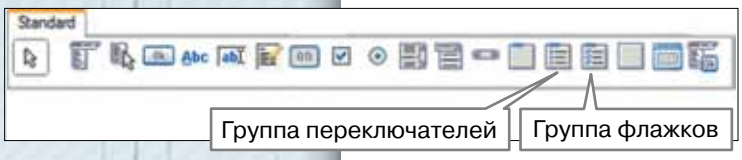


Рис. 24.9

Элементы управления `RadioGroup` и `CheckGroup` обладают своими специальными свойствами (табл. 24.9).

Таблица 24.9

Свойство	Описание
`Caption`	Заголовок группы
`Columns`	Количество столбцов в группе. По умолчанию — 1
`ItemIndex`	Определяет номер (начиная с 0) выделенного в группе элемента управления. Если не выделен ни один, то значение свойства равно –1
`Items`	Содержит список заголовков элементов группы. Для ввода заголовков открывается редактор, вызываемый с помощью кнопки ..., которая расположена справа в строке свойства *Items*

Чтобы добавить элемент управления *флажок* в программе, написанной на языке *Python* в среде *PyCharm*, после подключения модуля графического интерфейса `tkinter` и элемента управления *флажок* `from tkinter import ttk` следует описать объект *флажок*:

```
bool_var = tkinter.BooleanVar()
```

После чего разместить флажок на главной форме `main`:

```
check = ttk.Checkbutton(main, text = "Надпись", variable = bool_var)
check.pack()
```

Чтобы проверить, включён ли флажок, используют команду

```
bool_var.get()
```

ДЕЙСТВУЕМ

Упражнение 3. Заказ цветов.

Задание. Разработайте в среде *Lazarus* проект *Заказ цветов* для оформления электронного заказа цветов по образцу (рис. 24.10). В проекте нажатие кнопки *Оформить заказ* вызывает вывод соответствующего текста в текстовом поле, защищённом от изменений, а нажатие кнопки *Завершить* — закрывает окно формы.

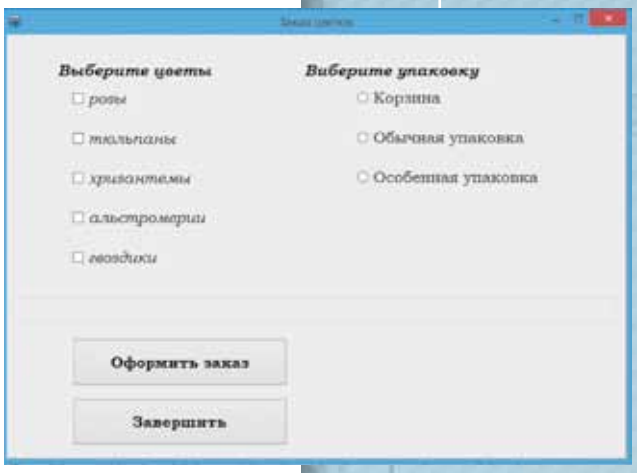


Рис. 24.10

1. Спланируйте проект. Продумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Заказ_цветов*.
3. Откройте среду *Lazarus*, создайте новый проект и сохраните его составляющие в папку *Заказ_цветов*. Измените значения свойств объекта `Form1`, разместите на форме нужные объекты и присвойте значения их свойствам для получения вида, как на рисунке 24.10. Учтите, что для всех объектов используется значение свойства шрифта — *Bookman Old Style*, размер — 14. Начертание определите по рисунку.
4. Создайте процедуру обработки события: нажата кнопка *Оформить заказ*. В окне редактора кода введите код (рис. 24.11).
5. Создайте процедуру обработки события нажатия кнопки *Завершить*. Для этого используйте метод `Close`.

```
procedure TForm.Button1Click(Sender: TObject);
var k1,k2,k3,k4,k5,p1,p2,p3: boolean;
      s, t: string;
begin
      k1 := CheckBox1.checked;
      k2 := CheckBox2.checked;
      k3 := CheckBox3.checked;
      k4 := CheckBox4.checked;
      k5 := CheckBox5.checked;
      p1 := RadioButton1.checked;
      p2 := RadioButton2.checked;
      p3 := RadioButton3.checked;
      s := '';
      if k1 then s := s+CheckBox1.Caption+' ';
      if k2 then s := s+CheckBox2.Caption+' ';
      if k3 then s := s+CheckBox3.Caption+' ';
      if k4 then s := s+CheckBox4.Caption+' ';
      if k5 then s := s+CheckBox5.Caption+' ';
      if p1 then t := ' в корзине';
      if p2 then t := ' в обычной упаковке;
      if p3 then t := ' в особенной упаковке;
      Edit1.Text := 'Вы выбрали '+s+t;

end;
```

Переменные *k1, k2, k3, k4, k5* определяют состояние элементов управления `CheckBox`

Переменные *p1, p2, p3* определяют состояние элементов управления `RadioButton`

В текстовой переменной *s* формируется сообщение о выборе цветов

В текстовой переменной *t* формируется сообщение об упаковке

Вывод результата в текстовое поле

Рис. 24.11

6. Запустите проект на выполнение. Проверьте, соответствуют ли условию задания действия, связанные с объектами управления экранной формы. При наличии ошибок — исправьте их.
7. Завершите работу с проектом и средой программирования.

Упражнение 4. Цветная форма.

Задание. Разработайте проект *Цветная форма* в среде *Lazarus*, в котором форма будет изменять значения своих свойств — цвет и размер в зависимости от выбранных переключателей в группах *Цвет* и *Размер*.

1. В папке *Учебные проекты* своей структуры папок создайте папку *Цветная_форма*.
2. Откройте среду *Lazarus,* создайте новый проект и сохраните его составляющие в папке *Цветная_форма*. Измените значение свойства `Caption` объекта `Form1` на *Цвета.*

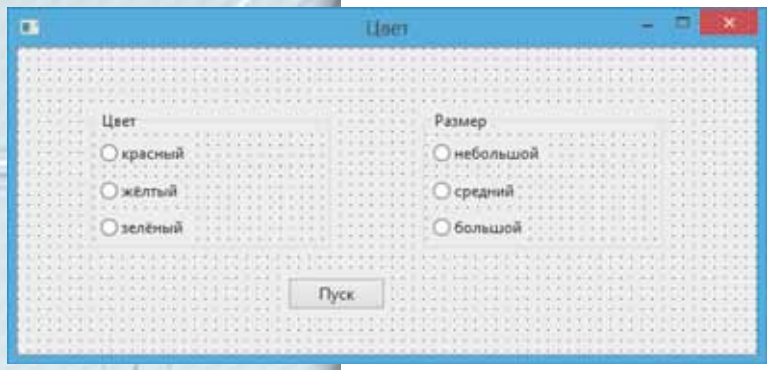


Рис. 24.12

3. Разместите на экранной форме нужные объекты — группы переключателей с заголовками *Цвет* и *Размер* (рис. 24.12).
4. Измените значения свойств `Items` каждой группы, введя соответствующие пояснения к переключателям (рис. 24.13).
5. Создайте процедуру обработки события: нажата кнопка *Пуск.*

В окне редактора кода введите команды для изменения значения свойства `Color` экранной формы в зависимости от включённого переключателя. Значение этого свойства начинается с букв `cl`, после которых записывают название цвета на английском языке. Например:

```
if RadioGroup1.ItemIndex=0 then form1.color:=clRed;
```

6. Введите команды изменения размера формы по образцу:

```
if RadioGroup2.ItemIndex=0 then
   begin
     form1.height:=100;
     form1.width:=400;
   end;
```

7. Для среднего размера выберите высоту 200, ширину 600, а для большого — 300, 800 соответственно.
8. Запустите проект на выполнение. Проверьте, соответствуют ли условию задания действия, связанные с объектами управления экранной формы. При наличии ошибок — исправьте их.
9. Завершите работу с проектом и средой программирования.

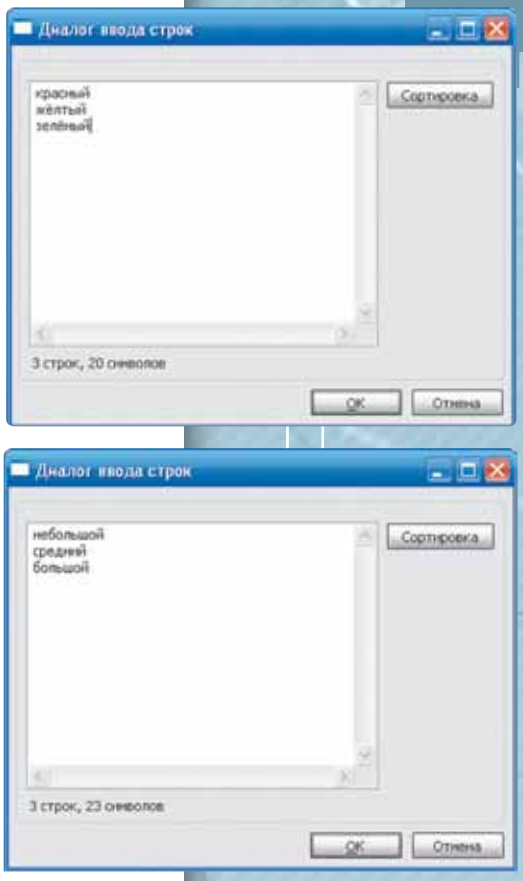


Рис. 24.13

Упражнение 5. Текст.

Задание. Разработайте в среде *PyCharm* проект, с помощью которого можно определить, состоит введённый текст из одного слова или из нескольких.

1. Откройте среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Текст* в папке *Учебные проекты* своей структуры папок.
3. В области программного кода запишите команды по образцу (рис. 24.14).
4. Запустите проект на выполнение. Подберите тестовые данные и проверьте правильность составленного программного кода.
5. Измените программный код так, чтобы текст можно было вводить в текстовое поле, а в окне сообщения — получать ответ. Объясните, какие команды из предложенных вы будете использовать в программном коде:

```
import tkinter
import tkinter.messagebox
main = tkinter.Tk()
str_var = tkinter.StringVar()
def button_click():
   s = str_var.get()
   if ' ' in s:
         tkinter.messagebox.showinfo("Результат",
         "введено несколько слов")
   else:
         tkinter.messagebox.showinfo("Результат",
         "введено одно слово")
label = tkinter.Label(text="Введите текст")
label.pack()
edit = tkinter.Entry(main, textvariable = str_var)
edit.pack()
button = tkinter.Button(main, text="Проверить",
command=button_click)
button.pack()
main.mainloop()
```

Проверка условия: пробел входит в строку *s*

```
s=input('введите текст')
if ' ' in s:
   print('введено несколько слов')
else:
    print('введено одно слово')
```

Рис. 24.14

6. Запустите проект на выполнение. Получите результат для введённого одного и нескольких слов. Завершите работу со средой программирования.

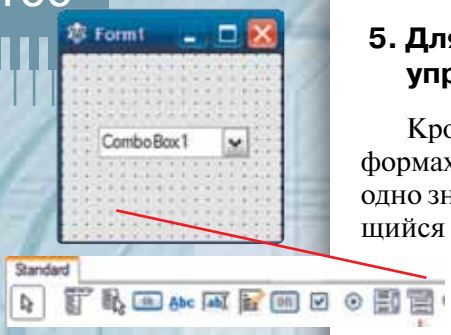


Рис. 24.15

5. Для чего на форме используют элемент управления *раскрывающийся список*?

Кроме текстового поля, содержащего только одно значение, на экранных формах размещают также элемент управления *список*, позволяющий выбрать одно значение из предложенного перечня. В среде *Lazarus* создать раскрывающийся список можно с помощью компонента `ComboBox` (рис. 24.15).

В отличие от других компонентов управления, которые вы уже изучали, список `ComboBox` имеет специальные свойства (табл. 24.10).

Таблица 24.10

Свойство	Описание
`Items`	Элементы списка
`Count`	Количество элементов списка
`Sorted`	Признак необходимой сортировки (если это свойство имеет значение `True`) после добавления очередного элемента списка
`ItemIndex`	Номер выбранного элемента (нумерация начинается с нуля. Если ни один из элементов не выбран, то значение равно –1)
`DropDownCount`	Количество элементов, отображающихся в списке. Чтобы отобразить остальные — используют полосу прокрутки

В программах на языке *Python* в среде *PyCharm* для того, чтобы использовать раскрывающийся список после подключения модуля графического интерфейса `tkinter` и элемента управления `from tkinter import ttk`, сначала описывают объект *список*:

```
choice_var = tkinter.StringVar()
```

Потом размещают список на главной форме `main`, задавая значения элементам списка:

```
choice = ttk.ComboBox(main, textvariable=choice_var)
choice['values'] = ('Значение1', 'Значение2', 'Значение3')
choice.pack()
```

Для определения выбранного элемента списка используют команду

```
choice _var.get()
```

Рис. 24.16

Упражнение 6. Карточка участника соревнований.

Задание. Разработайте проект *Карточка участника соревнований* в среде *PyCharm*, в котором пользователь вводит свою фамилию, имя и отчество в текстовое поле, помечает флажками вид соревнования, из раскрывающегося списка выбирает возрастную группу. После нажатия кнопки *Регистрация* (рис. 24.16) получает в окне сообщение о подтверждении регистрации.

1. Откройте среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Карточка* в папке *Учебные проекты* своей структуры папок.
3. Спланируйте, какие объекты нужно импортировать в программу. Выберите их из предложенного списка и добавьте к проекту в окне редактора кода:

```
import tkinter
from tkinter import ttk
from tkinter import messagebox
```

4. Определите, какие переменные из предложенного списка могут принимать значения *фамилия*, *имя и отчество*, *вид соревнования*, *возрастная группа*:

```
name_var = tkinter.StringVar()
bool_var1 = tkinter.BooleanVar()
bool_var2 = tkinter.BooleanVar()
bool_var3 = tkinter.BooleanVar()
choice_var = tkinter.StringVar()
```

5. Создайте объекты и разместите их на главной форме по образцу (рис. 24.16). Для создания объекта *раскрывающийся список* запишите команду:

```
choice['values'] = ('младшая', 'средняя', 'старшая')
```

6. Запишите код обработки события нажатия кнопки:

```
def button_click():
    s = ""
    s += name_var.get()+", вы выбрали: "
    if bool_var1.get():
        s += "бег на 100 м "
    if bool_var1.get():
        s += "прыжки в длину "
    if bool_var1.get():
        s += "челночный бег "
        s += " в возрастной категории " + choice_var.get()
    messagebox.showwarning("Регистрация", s)
```

7. Добавьте к программному коду команду запуска обработки событий программы:

```
main.mainloop()
```

8. Запустите программу на выполнение, проверьте, получили ли вы для предложенных значений (рис. 24.17, *а*) ожидаемый результат (рис. 24.17, *б*).

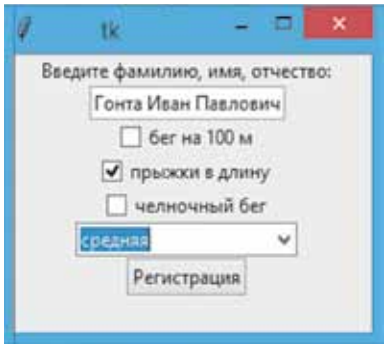


Рис. 24.17, *а*

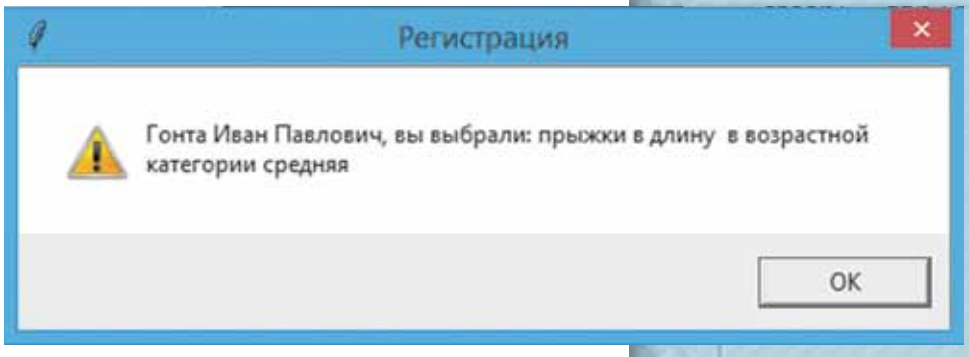
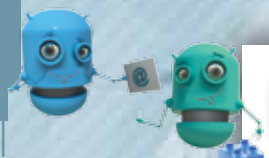


Рис. 24.17, *б*

9. Сохраните изменения в проекте. Закройте окно среды программирования.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 24* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.



РАБОТАЕМ В ПАРАХ

1. Составьте в паре пять выражений на сравнение значений величин разных типов на выбранном языке программирования. Предложите их проанализировать другой паре. Проверьте правильность полученных ответов.
2. Обсудите, чем будет отличаться порядок разработки проекта об агрегатном состоянии воды в среде *PyCharm* от его создания в среде *Lazarus*. Реализуйте этот проект.
3. Обсудите, достаточно ли известных вам средств языка программирования *Free Pascal* для того, чтобы определить, из скольких слов состоит некоторое предложение, вводимое в текстовое поле после запуска проекта на выполнение. Попробуйте реализовать этот проект в среде *Lazarus*. Какие преимущества языка программирования *Python* перед языком программирования *Free Pascal* вы нашли при этом?
4. Обсудите, можно ли реализовать проект *Карточка участника соревнований* в среде *Lazarus*. Определите, что целесообразнее использовать при разработке экранной формы — соответствующие элементы управления или их группы? Разработайте проект, распределив роли: один создаёт объекты экранной формы, а другой — пишет программный код обработки событий.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Определите, какое логическое значение принимает логическое выражение, записанное на языках программирования *Free Pascal* и *Python*:

Free Pascal	***Python***
1) `True and True`	1) `False and False`
2) `1 = 1 or 2 <> 1`	2) `1 == 1 and 2 == 1`
3) `'test' = 'test'`	3) `«test» == «testing»`
4) `'testing' <> 'test'`	4) `«test» != «testing»`
5) `False and 1 = 1`	5) `True and 0 != 0`
6) `not (10 = 1 or 1000 = 1000)`	6) `not (1 == 1 and 0 != 1)`

2. Известно такое соотношение идеального веса и роста человека в зависимости от возраста: от значения роста человека в сантиметрах берутся последние две цифры; если возраст человека до 25 лет, то его вес должен составлять на 5 кг меньше от полученного двузначного числа, для людей от 25 до 45 лет — быть равным этому двузначному числу, а для тех, кто старше 45 лет — двузначное число нужно увеличить на 5. В выбранной самостоятельно среде программирования разработайте проект *Идеальный вес*, в котором пользователь вводит в текстовое поле свой рост в сантиметрах и вес, выбирает диапазон возраста, к которому он относится. После нажатия кнопки запуска проекта в текстовом поле или окне сообщения получают вывод: идеальный вес, избыточный вес, недостаточный вес.
3. В выбранной самостоятельно среде программирования разработайте проект *Оценки*, в котором в текстовое поле выводится описание достигнутого уровня на основе школьной оценки, выбранной с помощью элемента управления (выберите самостоятельно — флажки, переключатели, раскрывающийся список):

1) 1, 2, 3 — начальный уровень;
2) 4, 5, 6 — средний уровень;
3) 7, 8, 9 — достаточный уровень;
4) 10, 11, 12 — высокий уровень.

4. В выбранной самостоятельно среде программирования разработайте проект *Мишень*, в котором по введённым в текстовые поля действительным числам x и y в окне сообщения будет выведена фраза «Поздравляем! Меткий выстрел!», если точка с координатами $(x; y)$ принадлежит заштрихованной области (рис. 24.18), и фраза «Жаль, не попал!» — в противном случае.

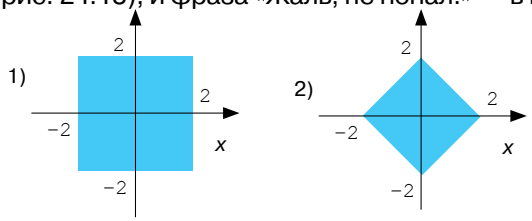


Рис. 24.18

5. В выбранной самостоятельно среде программирования разработайте проект *Правила дорожного движения*, по которому можно определить, не нарушил ли водитель, проехавший расстояние *s* за время *t*, правила дорожного движения, если на пути установлено одно из предложенных ограничений скорости: 40 км/ч, 60 км/ч, 90 км/ч. Учтите, что данные вводятся в текстовые поля, ограничения выбираются с помощью элементов управления (выберите самостоятельно), а результат — выводится в окне сообщения.

6. В среде программирования *Lazarus* разработайте проект *Радуга*, в котором после выбора цвета радуги и нажатия кнопки *Показать* открывается дополнительная экранная форма, цвет которой соответствует выбранному на главной форме.

7. В среде программирования *PyCharm* разработайте проект, в котором выполняется проверка, является ли данное слово палиндромом (можно прочитать слева направо так же, как справа налево). Например, слово «око» — это палиндром. Продумайте средства для того, чтобы слово «Анна» тоже было определено как палиндром, учитывая разницу между прописными и строчными буквами.

8. С помощью условного оператора на языках программирования *Python* и *Free Pascal* запишите фрагмент программного кода для:

1) замены значения переменной *s* действительного типа его абсолютной величиной;
2) присваивания переменной *k* значения 0, если её начальное значение принадлежит интервалу (0; 5);
3) присваивания переменной *t* значения `True`, если введённые координаты x и y принадлежат первой или третьей координатной четверти;
4) нахождения значения выражения $max^2(x, y) - min^2(x, y)$, где max — большее и min — меньшее из двух чисел x и y.

9. В выбранной самостоятельно среде программирования разработайте проект *Коврики*, в котором можно определить, какое максимальное количество квадратных ковриков со стороной c, где c — целое число, нужно использовать, чтобы застелить комнату с полом размером $a \times b$, где a и b — целые числа. В проекте также следует определить, какая площадь не будет накрыта ковриками, если их нельзя накладывать друг на друга или подгибать. Предусмотрите также ситуацию, когда размер коврика превышает размер пола комнаты.

25. РЕАЛИЗАЦИЯ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ НА ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

ВСПОМНИТЕ:

- с какой целью в алгоритмах используют структуру повторения;
- представление алгоритмической структуры повторения графическим способом;
- описание алгоритмов с определённым количеством повторений в среде *Скретч*;
- описание циклических алгоритмов с условием в среде *Скретч*;
- что такое вложенные циклы.

ВЫ УЗНАЕТЕ:

- как на языках программирования описывается цикл с предусловием;
- как на языках программирования описывают цикл со счётчиком;
- каковы особенности использования циклов в программах, описанных на языках программирования *Free Pascal* и *Python*;
- как задать случайное число.

Рис. 25.1

ИЗУЧАЕМ

1. Как на языках программирования описывается цикл с предусловием?

Вы знаете, что алгоритм, в котором предусматривается многократное выполнение одного и того же набора команд, называют **циклическим**. В циклических алгоритмах используют алгоритмическую структуру **повторения**. В языках программирования для описания структуры повторения используют **операторы цикла**.

Любой оператор цикла состоит из двух частей: **заголовка** и **тела**. В заголовке цикла записываются условия, при которых выполнение цикла будет продолжаться или завершится, а в теле цикла содержатся команды, выполнение которых должно повторяться (рис. 25.1).

В языках программирования *Free Pascal* и *Python* **цикл с предусловием** описывается оператором `while…` (табл. 25.1).

Таблица 25.1

Язык программирования	Описание	
Free Pascal	`while <логическое выражение> do <команда>;`	Сокращённая форма

	`while <логическое выражение> do`
	`  begin`
	`  <команда1>;`
	`  <команда2>;`
	`  …`
	`  <команда n>;`
	`  end;`
	тело цикла
		Полная форма

Продолжение таблицы 25.1

Язык программирования	Описание	
Python	`while <логическое выражение>:` `    <команда>`	Сокращённая форма

	`while <логическое выражение>:`
	`    <команда1>`
	`    <команда2>`
	`    …`
	`    <команда n>`
	`else:`
	`    <команда иначе>`
	тело цикла
		Полная форма

Сокращённая форма оператора `while` работает следующим образом. Определяется значение логического выражения. Если это значение `True`, то выполняется команда, записанная после служебного слова `do` (*Free Pascal*), или команда, выделенная отступами, которая входит в блок `while` (*Python*). Если значение логического выражения `False`, то цикл завершается и управление передаётся команде, записанной в программном коде после тела цикла. В теле цикла обязательно используют команду, которая будет изменять значение величины, используемой в логическом выражении.

Полная форма оператора `while` на языке программирования *Free Pascal* предусматривает, что после служебного слова `do` будут выполняться команды, записанные между операторными скобками `begin ... end`. А на языке программирования *Python*, кроме команд, входящих в блок `while`, используется конструкция `else :`, после которой в новой строке делают отступы на 4 символа и записывают команды, которые должны выполняться, если значение логического выражения `False`.

ДЕЙСТВУЕМ

Упражнение 1. Урожай.

Задание. Фермер выращивает новый сорт растений для кормления животных. Он узучил закономерность, что после каждого скашивания, осуществляемого через неделю (один раз в две недели) не более 5 раз, количество зелёной массы увеличивается по формуле: $s = s + (i - 1) \cdot i$, где i — номер недели; s — количество зелёной массы, при первом срезе $s = 1$. Разработайте проект в среде *PyCharm* для определения, успеет ли фермер собрать n единиц необходимых кормов.

1. Откройте среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Урожай* в папке *Учебные проекты* своей структуры папок.
3. Импортируйте объекты, необходимые для описания графического интерфейса программы (рис. 25.2).
4. Опишите объекты, которые вы будете использовать в проекте.
5. Запишите программный код обработки события нажатия кнопки *Определить*.

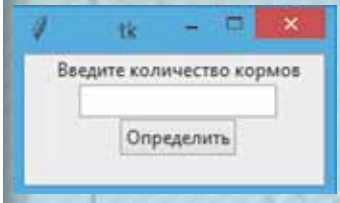


Рис. 25.2

```
def button_click():
        s = 1
        i=1

        st = str_var.get()
        k = int(st)
        while i <= 10:
            s += (i-1) * i
            i+=2

        if k <= s:
            tkinter.messagebox.showinfo("Результат", 'успеет
            получить '+ str(s))
        else:
            tkinter.messagebox.showinfo("Результат", 'Не успеет
            получить ')
```

6. Запишите программный код размещения объектов на главной форме:

```
label = tkinter.Label(text='Введите количество кормов:')
label.pack()
edit = tkinter.Entry(main, textvariable=str_var)
edit.pack()

button = tkinter.Button(main, text="Определить",
command=button_click)
button.pack()
```

7. Запишите программный код запуска событий главной формы.
8. Запустите проект на выполнение. Проверьте, получите ли вы для значения 50 сообщение «Успеет получить 141».
9. Завершите работу с проектом и средой программирования.

2. Как на языках программирования описывают цикл со счётчиком?

Для того чтобы описать повторение определённое количество раз, используют оператор **цикла со счётчиком** (рис. 25.3).

Рис. 25.3

В языках программирования *Free Pascal* и *Python* цикл со счётчиком описывается оператором `for ...` (табл. 25.2).

Таблица 25.2

Язык программирования	Описание

Free Pascal	параметр цикла; начальное значение параметра цикла; конечное значение параметра
	`for i:= <in> to <ik> do`
	`<команда>;`
	тело цикла; шаг изменения параметра цикла = 1
	`for i:= <in> downto <ik> do`
	`<команда>;`
	шаг изменения параметра цикла = –1

Продолжение таблицы 25.2

Язык программирования	Описание

Python	параметр цикла
	`for i in <диапазон>:`
	`<команда>`
	тело цикла

Величина параметра цикла, его начального и конечного значений могут быть целыми числами или принадлежать к некоторому списку.

Шаг изменения цикла всегда одинаков и равен интервалу между двумя ближайшими значениями типа параметра (при целочисленном значении параметра шаг равен 1).

Для определения диапазона значений параметра в программах на языке программирования *Python* можно использовать функцию `range` (табл. 25.3).

Таблица 25.3

Описание	Значение
`range(6)`	0, 1, 2, 3, 4, 5
`range(3,8)`	3, 4, 5, 6, 7
`range(7,16,2)`	7, 9, 11, 13, 15

Можно использовать величину перечисляемого типа. Например, список нечётных чисел первого десятка: 1, 3, 5, 7, 9 или список гласных букв русского алфавита 'а','о','у','е','ё','ы','и','э','ю','я'. Список значений в языке программирования *Free Pascal* заключают в круглые скобки (), а в *Python* — в квадратные [].

Если тело цикла состоит более чем из одной команды, как и в цикле с предусловием, в языке программирования *Free Pascal* используют операторные скобки `begin… end`.

Цикл `for`... выполняется по следующему алгоритму:

1. Параметру цикла `i` присваивается начальное значение.

2. Если значение параметра цикла больше, чем его конечное значение, то цикл завершается (в случае цикла со служебным словом `downto` в языке программирования *Free Pascal* цикл завершается, когда значение параметра цикла меньше, чем его конечное значение). Иначе выполняется п. 3.

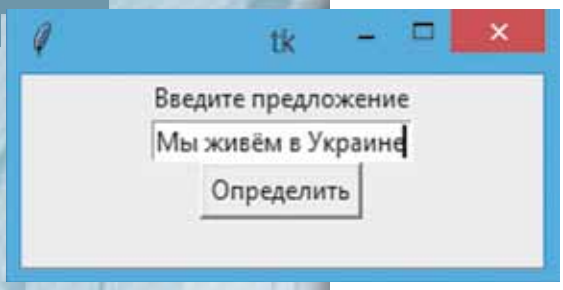
3. Выполняется команда.

4. Значение параметра цикла `i` меняется на соответствующий шаг, и осуществляется переход к п. 2 и т. д.

Таким образом, в отличие от оператора цикла `while`, в операторе цикла со счётчиком изменение значения счётчика осуществляется автоматически.

ДЕЙСТВУЕМ

Упражнение 2. Количество слов в предложении.
Задание. Разработайте в среде *PyCharm* проект, с помощью которого можно определить, сколько слов введено в текстовое поле, если известно, что между словами содержится только один пробел.



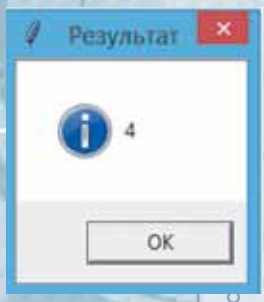


Рис. 25.4

1. Откройте среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Количество_слов* в папке *Учебные проекты* своей структуры папок.
3. Импортируйте объекты, необходимые для описания графического интерфейса программы (рис. 25.4).
4. Объявите и опишите величины `main` и `str_var`.
5. Запишите программный код обработки события нажатия кнопки *Определить:*

```
def button_click():
    s = 0
    z = ' '
    st = str_var.get()
    k = len(st)
    for i in range(0, k):
      if z == st[i]:
        s = s + 1
    tkinter.messagebox.showinfo("Результат", str(s+1))
```

6. Запишите в программном коде команды для создания объектов на форме `main` и получения значений переменных.
7. Запишите команду запуска событий на форме.
8. Запустите проект на выполнение. Проверьте, соответствует ли результат для введённого предложения рисунку 25.4.
9. Завершите работу с проектом и средой программирования.

3. Каковы особенности использования циклов в программах, написанных на языках программирования *Free Pascal* и *Python*?

В языке программирования *Free Pascal* реализован ещё один цикл — **цикл с постусловием**:

```
repeat
   <команды>;
until <логическое выражение прекращения цикла>;
```

В отличие от цикла с предусловием, в котором проверка истинности условия осуществляется до выполнения команд в теле цикла, в цикле с постусловием истинность условия проверяется после выполнения команд тела цикла. При этом если значение логического выражения `False`, то команды цикла повторяются, а если `True`, то выполнение цикла прекращается. Таким образом, цикл с постусловием будет выполнен хотя бы один раз, в отличие от цикла с предусловием, который может быть не выполнен ни разу, если логическое выражение сразу имеет значение `False`. Если в теле цикла нужно использовать более одной команды, то операторные скобки использовать не нужно — их роль выполняют служебные слова `repeat` и `until`, входящие в конструкции оператора цикла.

В языке программирования *Python* можно использовать досрочный выход из цикла. С помощью функции `break` можно прервать выполнение команд тела цикла и передать управление команде, которая следует после блока команд, относящихся к `while` или `for`.

ДЕЙСТВУЕМ

Упражнение 3. Простое число.

Задание. Число называется простым, если оно делится без остатка только на единицу и на себя. Разработайте программный проект в среде *Lazarus*, с помощью которого будет проверяться, является ли простым введённое в текстовое поле число (рис. 25.5).

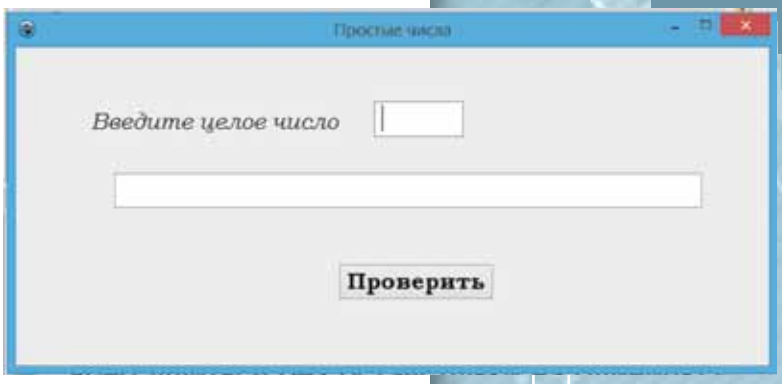


Рис. 25.5

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты своей структуры* папок создайте папку *Простое_число*.
3. Откройте среду *Lazarus*, создайте новый проект и сохраните его составляющие в папку *Простое_число*. Измените значения свойств объекта `Form1`, разместите на форме нужные объекты и задайте значения их свойствам, чтобы получить форму, как на рисунке 25.5.

```
var n, i: integer;
begin
n := strToint(edit1.Text);
i := 1;
repeat
     i := i+1
until n mod i=0;
if n = i then edit2.text := 'введённое число простое'
       else edit2.text := 'введённое число составное';
end;
```

Рис. 25.6

4. В программном коде обработки события нажатия кнопки *Проверить* введите команды (рис. 25.6).
5. Запустите проект на выполнение. Введите число 15. Проверьте полученный результат.
6. Запустите проект на выполнение ещё раз для значения 19. Проверьте полученный результат.
7. Сохраните изменения в проекте. Завершите работу с проектом и средой программирования.

4. Как задать случайное число?

Составляя алгоритмы в учебной среде *Скретч*, вы использовали команду, с помощью которой получали для значений параметров некоторых команд случайные числа: координаты размещения объекта на сцене, цвет рисунка и т. п. (рис. 25.7).

Рис. 25.7

Случайные значения величин часто используют при составлении программ, в том числе игровых.

Для того чтобы в программе на языке программирования *Python* использовать значение, которое передаётся в программный код случайно, подключают модуль `random`:

```
import random
```

Тогда в программном коде можно получить случайные значения величин различных типов (табл. 25.4).

Таблица 25.4

Описание	Значение
`random.random()`	Случайное действительное число от 0,0 включительно до 1,0
`random.randint(1,10)`	Случайное целое число от 1 до 10 включительно
`random.choice (['а','о','у', 'е','ё','ы','и','э','ю','я'])`	Гласная буква русского алфавита, выбранная случайно

В программах, написанных на языке программирования *Free Pascal*, в программном коде обработки события один раз обращаются к функции `Randomize`, которая подключает генератор случайных чисел и позволяет получать каждый раз другие значения. Тогда в программном коде можно получить случайные значения величин различных типов (табл. 25.5).

Таблица 25.5

Описание	Значение переменной A
`Randomize;` `A:=random;`	Случайное действительное число от 0,0 включительно до 1,0
`Randomize;` `A:=random(10);`	Случайное целое число от 0 до 10 включительно
`Randomize;` `A:=random(b-a+1)+a;`	Случайное целое число от a до b включительно

ДЕЙСТВУЕМ

Упражнение 4. Игра.

Задание. Составьте программу на языке программирования *Python*, которая реализует игру *Отгадай число* между компьютером и пользователем. Пользователь не более чем за 6 попыток должен отгадать «задуманное» число из диапазона от 1 до 20. Если пользователь отгадает, то на экран выводится сообщение о количестве попыток, если нет — выводится «задуманное» число.

1. Откройте среду программирования *PyCharm*.
2. Создайте новый файл программы на языке *Python* с именем *Отгадай число* в папке *Учебные проекты* своей структуры папок.
3. В программном коде запишите команды по образцу. Проанализируйте назначение каждой строки кода.

```
import random
guessesTaken = 0
print('Привет! Как тебя зовут?')
myName = input ()
number = random.randint(1, 20)
print('Итак, ' + myName + ', Я задумал число от 1 до 20.')
while guessesTaken < 6:
   print('Попробуй отгадать.')
   guess = input ()
```

```
    guess = int(guess)
    guessesTaken = guessesTaken + 1
    if guess < number:
      print('Твоё число меньше задуманного.')
    if guess > number:
      print('Твоё число больше задуманного.')
    if guess == number:
      break
if guess == number:
    guessesTaken = str(guessesTaken)
     print('Очень хорошо, ' + myName + '! Для отгадывания
     тебе понадобилось ' + guessesTaken + ' попыток!')
if guess != number:
    number = str(number)
    print('Нет. Число, которое я задумал, - ' + number)
```

4. Запустите проект на выполнение. Попробуйте выиграть в игре. Завершите работу со средой программирования.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 25* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Обсудите, как составить схему для обобщения описания циклов на языках программирования *Free Pascal* и *Python*. Создайте такую схему средствами текстового процессора для выбранного языка программирования. Сравните созданные схемы со схемами, выполненными другой парой учеников.
2. Обсудите, чем будет отличаться проект *Урожай*, разработанный на языке программирования *Python*, от аналогичного, разработанного на языке программирования *Free Pascal*. Реализуйте такой проект в среде *Lazarus*.
3. Обсудите, чем будет отличаться проект *Простое_число*, разработанный на языке программирования *Free Pascal*, от аналогичного, разработанного на языке программирования *Python*. Реализуйте такой проект в среде *PyCharm*.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Какие значения будут присвоены переменным, имена которых встречаются в командах, в результате выполнения фрагментов программ (для различных значений исходных данных), представленных в таблице 25.6.

Таблица 25.6

Язык программирования	Описание фрагмента программы	Исходные данные

Free Pascal	1)
	`i := n;`
	`while i < 5 do`
	`  begin`
	`  x := x + 1;`
	`  i := i + 1`
	`  end;`
		при 1) n = 6; x = 0
		2) n = 2; x = 0
		3) n = 4; x = 0

	2)
	`for i := 1 to 5 do`
	`  begin`
	`  n := n + 2;`
	`  s := s + n`
	`  end;`
		при 1) n = 1; s = 0
		2) n = 2; s = 0
		3) n = 4; s = –1

Python	3)
	`i := 1; x := 0;`
	`while (i < n) and (y = 'ДА'):`
	`  x := x + 1;`
	`  i := i + 1`
		при 1) n = 3; y = 'ДА'
		2) n = 3; y = 'НЕТ'
		3) n = 1; y = 'ДА'
		4) n = 1; y = 'НЕТ'
		5) n = 5; y = 'ДА'

	4)
	`for i in range(1, 5):`
	`  for j in range(1, 5):`
	`  r = i * j`
	`  print(i + ', умноженное`
	`  на ' + j + ' равно' + r);`	

2. Запишите фрагменты программ на языках программирования *Free Pascal* и *Python* для выполнения следующих действий:

1) печать таблицы умножения на 9;
2) печать значений квадратов чисел от 10 до 1;
3) вычисление суммы целых чисел от 1 до 10;
4) вычисление произведения чётных чисел первого десятка;
5) вычисление суммы дробей с числителем 1, знаменателями которых являются нечётные числа от 3 до 9;
6) определение количества введённых предложений (предложение заканчивается точкой, восклицательным или вопросительным знаком).

3. Разработайте проект в выбранной среде программирования, с помощью которого можно определить наименьший общий делитель двух целых чисел.

4. Разработайте проект в выбранной среде программирования, в котором по введённым начальным и конечным значениям температуры и шагу их изменения в текстовое поле выводится список соответствующих температур по шкале Фаренгейта, если для такого перевода используют формулу:

$$T_f = \frac{9}{5} T_c + 32.$$

5. Разработайте проект в выбранной среде программирования, в котором для выражения, содержащего арифметические операции без скобок и математических функций, в сообщении будет выведено, какие действия нужно выполнить.

6. Разработайте проект в выбранной среде программирования, в котором из текста, вводимого с клавиатуры, будет выведена его часть, расположенная: 1) до первой точки; 2) от второй точки и до конца.

26. ПРАКТИЧЕСКАЯ РАБОТА 12

СОСТАВЛЕНИЕ И ВЫПОЛНЕНИЕ РАЗВЕТВЛЯЮЩИХСЯ И ЦИКЛИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РАБОТЫ С ВЕЛИЧИНАМИ

ВСПОМНИТЕ

- Как описать алгоритмическую структуру ветвления на языках программирования *Free Pascal* и *Python*;
- как описать алгоритмическую структуру повторения на языках программирования *Free Pascal* и *Python*;
- как описать операции с величинами различных типов на языках программирования *Free Pascal* и *Python*;
- для чего на электронных формах используют элементы управления *переключатель*, *флажок*, *список*.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 12.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Расчёт оплаты за электроэнергию (*16 баллов*)

В Украине за потребление электроэнергии определён следующий порядок оплаты в зависимости от потребляемого объёма: стоимость 1 кВт при потреблении электроэнергии до 100 кВт в месяц — 45,6 грн, от 100 кВт до 600 кВт — 78,9 грн, свыше 600 кВт — 147,9 грн.

Некоторым потребителям предоставляется льготный объём бесплатного пользования электроэнергией, например 30 кВт. Если такая льгота есть, то её вычитают из потреблённого объёма, а далее расчёт осуществляется по принятым тарифам.

Разработайте в выбранной среде программирования проект *Оплата за электроэнергию*, в котором пользователь вносит в текстовое поле объём потребляемой электроэнергии и обозначает с помощью элементов управления (определите самостоятельно) наличие льготы. После нажатия кнопки *Рассчитать* получает в окне сообщения размер суммы для оплаты за электроэнергию.

Задание 2. Пенсионный калькулятор (*19 баллов*)

Мама восьмиклассника Максима работает в отделении Пенсионного фонда. Максим предложил разработать для неё программу, которую она может использовать в своей работе.

В Украине был определён следующий порядок налогообложения пенсии физических лиц:

- устанавливается минимальная заработная плата (например, с января по апрель — 1378 грн, с мая по ноябрь — 1450 грн, с декабря — 1550 грн);
- если сумма начисления s меньше, чем три минимальные заработные платы, то пенсия не облагается налогом;

- если сумма начисления s больше трёх минимальных заработных плат, но не превышает 10 минимальных заработных плат, то с суммы, превышающей три минимальные заработные платы, высчитывают 15 % налога;
- если сумма начисления s больше, чем 10 минимальных заработных плат, то облагается сумма, превышающая три минимальные зарплаты. С оставшейся суммы до 10 минимальных заработных плат высчитывают 15 % налога, а с суммы, превышающей 10 минимальных заработных плат, высчитывают 20 % налога.

Разработайте в выбранной среде программирования проект *Пенсионный калькулятор*, в котором пользователь вносит в текстовое поле размер начисленной пенсии и выбирает с помощью элементов управления (определите самостоятельно) размер минимальной заработной платы. После нажатия кнопки *Рассчитать* в текстовых полях, защищённых от изменений, получает размеры суммы начисленной пенсии и налога.

Задание 3. Сумасшедшие скидки (*16 баллов*)

В магазине для некоторых товаров (фрукты, овощи, печенье) приняли систему скидок: товар, который не продан за неделю, на следующей неделе дешевеет на 10 %, ещё через неделю — на 20 % и т. д.

Разработайте в выбранной среде программирования проект *Скидки*, в котором пользователь выбирает из списка продукт и в текстовом поле вносит номер недели покупки. После нажатия кнопки *Рассчитать* в окне сообщения получает цену товара на дату покупки. Учитывайте, что проценты скидки рассчитываются, начиная со второй недели, по формуле:

$$price2 = price1 \cdot \frac{100 - 1 \cdot 10}{100}$$

$$price3 = price2 \cdot \frac{100 - 2 \cdot 10}{100}$$

...

Задание 4. Кролеферма (*18 баллов*)

Ваш дедушка в деревне решил разводить кроликов. Он узнал, что еще в XIII в. Леонардо Пизанский вывел формулу закона размножения кроликов, которая определяет ряд чисел, получивших название чисел Фибоначчи: 1, 1, 2, 3, 5, 8 ... Следует заметить, что, начиная с третьего числа, каждое следующее равно сумме двух предыдущих, то есть имеет место формула:

$$F_n = F_{n-1} + F_{n-2}.$$

Разработайте проект в выбранной среде программирования, в котором после нажатия кнопки *Рассчитать* по введённому в текстовое поле порядковому номеру, определяющему последовательность этапа размножения кроликов, в окне сообщения отображается соответствующее число Фибоначчи.

Задание 5. Сравнение (*12 баллов*)

Члены математического кружка ознакомились с понятием факториала числа — произведения натуральных чисел от 1 до данного числа: $n!=1 \cdot 2 \cdot 3 \cdot \ldots n$.

Разработайте проект в выбранной среде программирования, с помощью которого можно будет сравнить значение факториала числа и его квадрата.

27. ГРАФИЧЕСКОЕ ОТОБРАЖЕНИЕ ДАННЫХ В ЯЗЫКАХ ПРОГРАММИРОВАНИЯ

ВСПОМНИТЕ:

- какие инструменты рисования используют в среде графического редактора;
- как создают рисунки и форматируют их в среде текстового процессора;
- чем отличаются рисунки, созданные в графическом редакторе и текстовом процессоре;
- какие средства для рисования используются в учебной среде создания и выполнения алгоритмов *Скретч*.

ВЫ УЗНАЕТЕ:

- как в среде программирования *Lazarus* используются рисунки, находящиеся во внешних файлах;
- как в среде *Lazarus* на форму добавляют графические фигуры;
- как в среде *Lazarus* рисуют линию, сектор и ломаную;
- как построить изображение в среде программирования *PyCharm*.

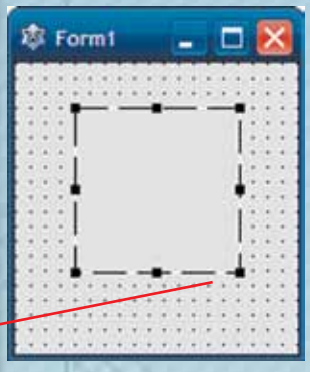
ИЗУЧАЕМ

1. Как в среде программирования *Lazarus* используются рисунки, находящиеся во внешних файлах?

В программах часто используются рисунки: иллюстрации, движущиеся изображения, фоны и т. п. В некоторых средах программирования существуют средства, обеспечивающие добавление готовых изображений к программному коду или создание и форматирование рисунков в самой программе. Например, в учебной среде создания и выполнения алгоритмов *Скретч* вы использовали такие средства для работы с графикой:

- меняли образы объектов, загружая их из библиотеки;
- рисовали объекты во встроенном графическом редакторе;
- выполняли построение изображений исполнителем, указывая в программном коде команды группы *Перо*.

Среда программирования *Lazarus* содержит средства для работы с графическими изображениями. На формах можно размещать рисунки, хранящиеся во внешних файлах. Компоненты экранной формы, используемые для размещения графических объектов, находятся на панели компонентов *Additional* (*Дополнительная*). Для вставки контейнера, в который добавляется рисунок из файла, на форме размещают компонент `Image` (рис. 27.1).



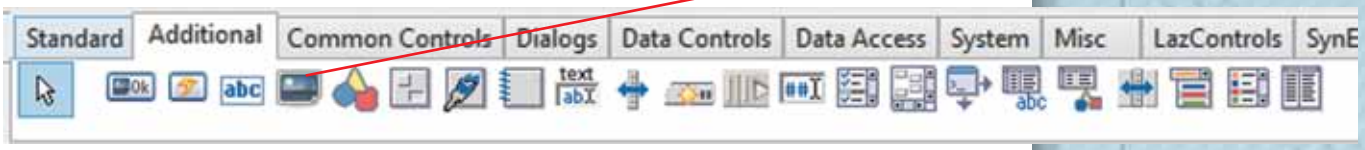


Рис. 27.1

Основные свойства компонента `Image` представлены в таблице 27.1.

Таблица 27.1

Свойство	Описание
`Picture`	Графическое изображение, отображаемое в поле компонента
`Width, Height`	Размер компонента. Если размер компонента меньше, чем размер изображения, и значения свойств `AutoSize`, `Strech` и `Proportional` равны `False`, то отображается часть изображения
`Proportional`	Автоматическое масштабирование картинки с сохранением пропорций, без искажения (значение `True`)
`Strech`	Автоматическое масштабирование (сжатие или растяжение) изображения в соответствии с реальным размером компонента. Если размер компонента не пропорционален размеру изображения, то изображение будет искажено
`AutoSize`	Автоматическое изменение размера компонента до реального размера изображения
`Visible`	Отображение изображения на форме
`Canvas`	Поверхность, на которую можно вывести изображение

Для того чтобы изменить значение свойства `Picture`, используют алгоритм добавления изображения на форму:

1. Вызвать окно загрузки графического изображения в строке этого свойства в окне *Инспектора объектов* (рис. 27.2).
2. Нажать кнопку *Загрузить* в окне *Диалог загрузки изображения*. В окне, которое открывается при этом, выбрать нужный графический файл и нажать кнопку *Открыть*. Далее в окне просмотра отображается содержимое этого файла (рис. 27.3).
3. Нажать кнопку *OK* и перейти к заданию значений свойствам компонента (рис. 27.4).

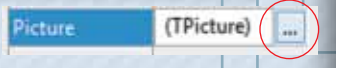


Рис. 27.2

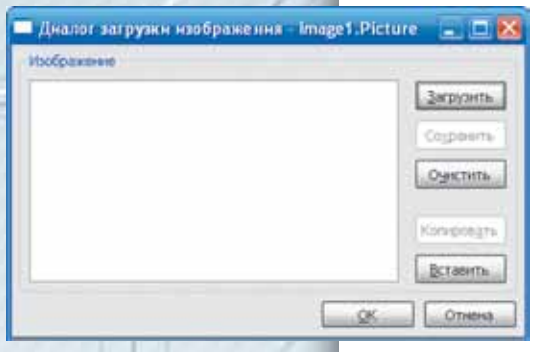
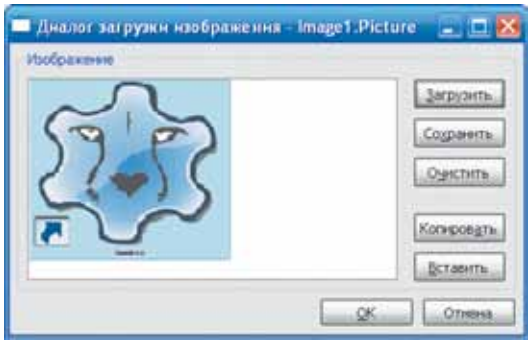


Рис. 27.3



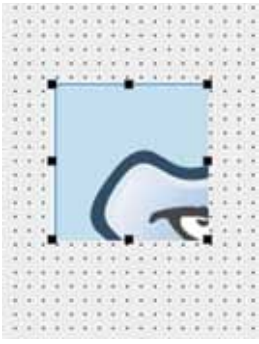


Рис. 27.4

ДЕЙСТВУЕМ

Упражнение 1. Озон.

Задание. В среде программирования *Lazarus* создайте проект, форма которого содержит рисунок, находящийся в файле *Ozon.jpg* в папке *Программирование*. При нажатии кнопки *Увеличить* размеры рисунка увеличиваются (имитируется эффект приближения, рис. 27.5).

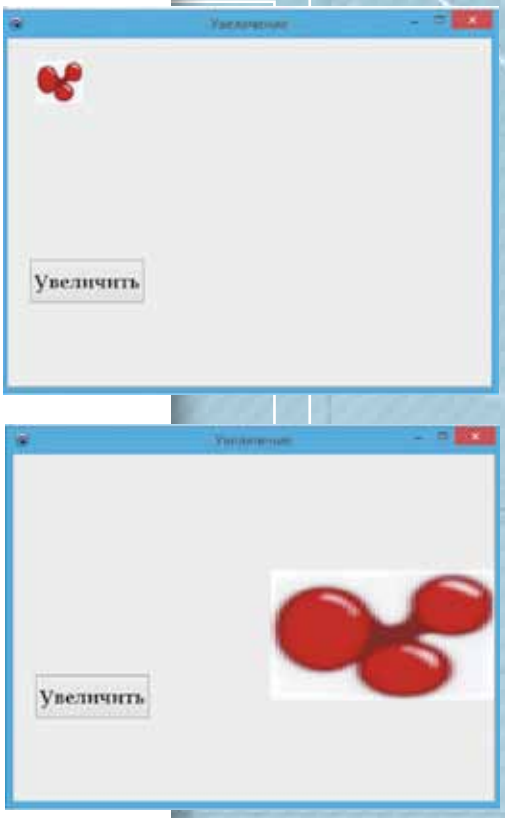


Рис. 27.5

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Озон*.
3. Откройте среду *Lazarus*, создайте новый проект. Измените значения свойств объекта *Form1*, разместите на форме объекты и задайте значения их свойствам (табл. 27.2).

Таблица 27.2

Объект	Свойство	Значение
Form1	Caption	Увеличение
Button1	Caption	Увеличить
	Font	Bookman Old Style, жирный, 16
Image1	Picture	*Ozon.jpg*
	Strech	True

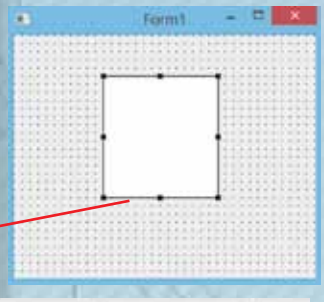
4. Создайте процедуру обработки события нажатия кнопки *Увеличить*. В окне редактора кода введите команды для перемещения рисунка и увеличения его:

```
Image1.Top := Image1.Top + 100;
Image1.Left := Image1.Left + 250;
Image1.Height := Image1.Height + 100;
Image1.Width := Image1.Width + 100;
```

5. Запустите проект на выполнение. Проверьте, получили ли вы требуемый результат.
6. Завершите работу с проектом и средой программирования.

2. Как в среде *Lazarus* на форму добавляют графические фигуры?

В среде *Lazarus* можно размещать некоторые геометрические фигуры на форме с помощью компонентов или создать программный код для рисования фигур в процессе выполнения программы. Для размещения на форме фигур, а именно: прямоугольника, эллипса, треугольника, ромба и т. д. — используют компонент *Shape* панели компонентов *Additional* (*Дополнительная*). По умолчанию на экранной форме будет размещён прямоугольник (рис. 27.6).



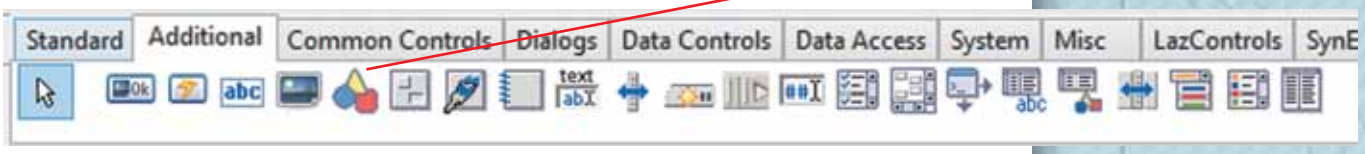


Рис. 27.6

Чтобы изменить форму фигуры, используют свойство `Shape` (рис. 27.7).

Чтобы изменить цвет фигуры и стиль заливки, используют свойство `Brush`.

Например, `Shape1.Brush.Color:=clRed` — цвет объекта `Shape1` будет красным, `Shape1.Brush.Style:=bsSolid` — сплошная заливка.

В среде *Lazarus* свойство `Color` может принимать фиксированный набор значений (табл. 27.3).

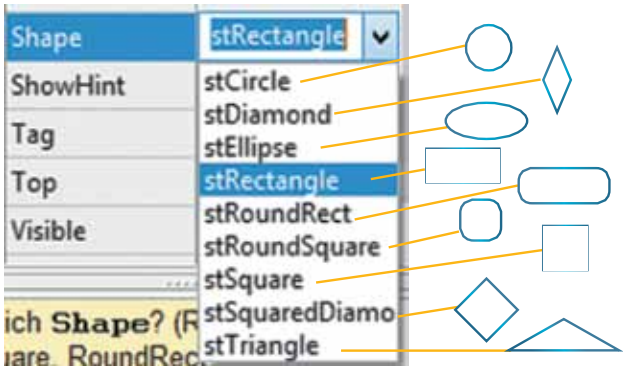


Рис. 27.7

Таблица 27.3

Значение	Цвет	Значение	Цвет	Значение	Цвет
`clBlack`	чёрный	`clSilver`	серебряный	`clPurple`	розовый
`clMaroon`	каштановый	`clRed`	красный	`clTeal`	пурпурный
`clGreen`	зелёный	`clLime`	салатовый	`clGray`	серый
`clOlive`	оливковый	`clBlue`	синий	`clAqua`	бирюзовый
`clNavy`	тёмно-синий	`clYellow`	жёлтый	`clWhite`	белый

Свойство `Style` может принимать значения из перечня, указанного в таблице 27.4.

Таблица 27.4

Значение	Описание	Значение	Описание
`bsSolid`	Сплошная заливка	`bsHorizontal`	Горизонтальная штриховка
`bsClear`	Без заливки	`bsVertical`	Вертикальная штриховка
`bsFDiagonal`	Диагональная штриховка с наклоном линий вперёд	`bsBDiagonal`	Диагональная штриховка с наклоном линий назад
`bsCross`	Горизонтально-вертикальная штриховка в клеточку	`bsDiagCross`	Диагональная штриховка в клеточку

К объектам `Shape` чаще всего применяют события перемещения мыши `OnMouseMove`, нажатия и отпускания кнопки мыши `OnMouseDown` и `OnMouseUp`. Каждое из двух последних событий происходит, если щёлкнуть мышью на фигуре, размещённой на форме, — в этом случае происходит и нажатие, и отпускание кнопки мыши.

ДЕЙСТВУЕМ

Упражнение 2. Преобразование.

Задание. Разработайте проект в среде *Lazarus*, в котором при щелчке мышью на картинке круга он окрашивается в цвет, выбранный в группе переключателей *Цвет*, а при щелчке мышью на изображении квадрата — он окрашивается выбранным цветом и применяется стиль заливки, который выбирают в группе *Заливка* (рис. 27.8).

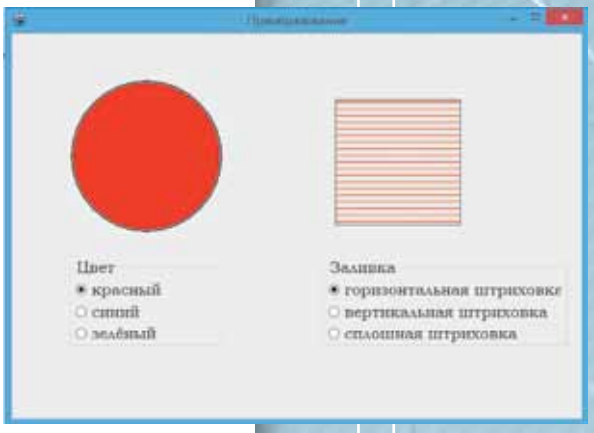


Рис. 27.8

1. Спланируйте проект. Подумайте, какие объекты будут использованы на экранной форме и какие события будут с ними происходить.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Преобразование*.
3. Откройте среду *Lazarus*, создайте новый проект. Измените значения свойств объекта *Form1*, разместите на форме объекты и задайте значения их свойствам (табл. 27.5).

Таблица 27.5

Объект	Свойство	Значение
Form1	Caption	Преобразование
Shape1	Shape	stCircle
Shape2	Shape	stSquare
RadioGroup1	Items	красный, синий, зелёный
RadioGroup2	Items	горизонтальная штриховка, вертикальная штриховка, сплошная заливка

4. Создайте процедуру обработки события щелчка кнопкой мыши на круге. Для этого выберите в таблице окна *Инспектора объектов* вкладку *События*, дважды щёлкните в ячейке справа от `OnMouseDown`. В окне редактора кода запишите программный код (рис. 27.9).

```
procedure TForm1.Shape1MouseDown(Seder: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: integer);
begin
    if RadioGroup1.ItemIndex=0 then Shape1.Brush.Color:=clRed;
    if RadioGroup1.ItemIndex=1 then Shape1.Brush.Color:=clBlue;
    if RadioGroup1.ItemIndex=2 then Shape1.Brush.Color:=clGreen;
end;
```

Рис. 27.9

5. Создайте процедуру обработки события *щелчок кнопкой мыши на квадрате*.
В окне редактора кода запишите программный код (рис. 27.10).

```
procedure TForm1.Shape2MouseDown(Seder: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: integer);
begin
     if RadioGroup1.ItemIndex=0 then Shape2.Brush.Color := clRed;
     if RadioGroup1.ItemIndex=1 then Shape2.Brush.Color := clBlue;
     if RadioGroup1.ItemIndex=2 then Shape2.Brush.Color  := clGreen;
     if RadioGroup2.ItemIndex=0 then Shape2.Brush.Style := bsHorizontal;
     if RadioGroup2.ItemIndex=1 then Shape2.Brush.Style := bsVertical;
     if RadioGroup2.ItemIndex=2 then Shape2.Brush.Style := bsSolid;
end;
```

Рис. 27.10

6. Запустите проект на выполнение. Проверьте, меняют ли свои свойства геометрические фигуры согласно условию задачи и выбранным значениям свойств.
7. Завершите работу с проектом и средой программирования.

3. Как в среде *Lazarus* рисуют линию, сектор и ломаную?

Для рисования линий и фигур при выполнении программы на форме в среде программирования *Lazarus* нужно размещать холст (полотно). Для этого обращаются к свойству `Canvas` — свойству объекта `Form`, который в свою очередь является объектом со своими методами и свойствами. Методы используют для отображения на холсте различных фигур, а свойства — для указания цвета линии, её толщины и т. п.

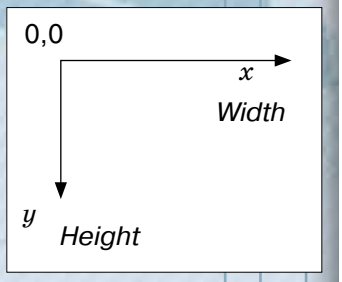


Рис. 27.11

Размер холста определяется значениями свойств `Height` — количество точек по вертикали и `Width` — по горизонтали. Холст состоит из отдельных точек — пикселей, координаты которых задаются значениями *x* и *y* (рис. 27.11).

Как и в графическом редакторе, в среде программирования *Lazarus* можно рисовать карандашом и кистью. Для этого используют свойства холста `Pen` и `Brush`:

`Form1.Canvas.Pen` или `Form1.Canvas.Brush`

Можно задавать значения свойств инструментов рисования:

`Form1.Canvas.Pen.Color` — цвет (см. табл. 27.3);

`Form1.Canvas.Pen.Width` — ширина (целое число);

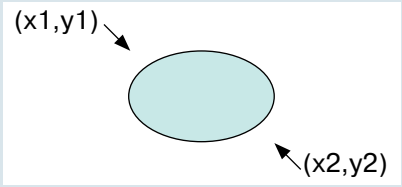
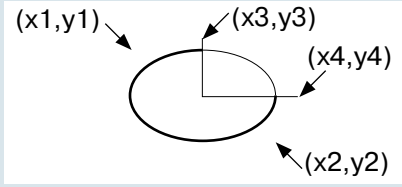
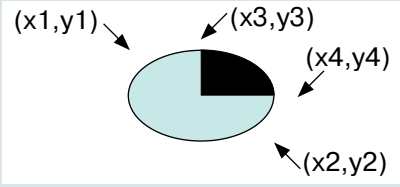
`Form1.Canvas.Pen.Style` — стиль (табл. 27.6).

Таблица 27.6

Значение	Вид	Значение	Вид
psSolid	————	psDot	- - - - - - - - -
psDash	— — —	psDashDot	—·—·—·—·—·—
psDashDotDot	—··—··—··—	psClear	Не отображается

Рисовать можно различные объекты: линию, ломаную, эллипс, сектор. Для этого используют соответствующие методы (табл. 27.7). Цвет, толщина и стиль этих объектов определяются значениями свойства `Pen`, а цвет и стиль заливки замкнутых фигур — значениями свойства `Brush`.

Таблица 27.7

Описание	Пояснение
`Form1.Canvas.LineTo(x,y)`	Построение линии до точки с координатами x, y
`Form1.Canvas.MoveTo(x,y)`	Перемещение карандаша в точку с координатами x, y
`Form1.Canvas.Restangle(x1,y1,x2,y2)`	Прямоугольник с верхним левым углом $x1$, $y1$ и нижним правым $x2$, $y2$
`Form1.Canvas.Polyline(x1,y1,x2,y2,x3,y3)`	Рисование ломаной линии с координатами $x1$, $y1$, $x2$, $y2$, $x3$, $y3$
`Form1.Canvas.Ellipse(x1,y1,x2,y2)`	Построение эллипса 
`Form1.Canvas.Arc(x1,y1,x2,y2,x3,y3,x4,y4)`	Построение дуги 
`Form1.Canvas.Pie(x1,y1,x2,y2,x3,y3,x4,y4)`	Построение сектора 

ДЕЙСТВУЕМ

Упражнение 3. Рисунок из линий.

Задание. В среде программирования *Lazarus* разработайте проект *Линии*, в котором на форме будут рисоваться линии с помощью протягивания мышью: начало линии будет расположено в позиции, в которой нажали кнопку мыши, а конец — там, где отпустили кнопку мыши.

1. Спланируйте проект. Подумайте, какие события будут происходить на экранной форме и какими средствами языка программирования *Free Pascal* их можно реализовать.
2. В папке *Учебные проекты* своей структуры папок создайте папку *Линии.*
3. Откройте среду *Lazarus*, создайте новый проект. Измените значение свойства `Caption` объекта *`Form1`* на *Рисование линий.*
4. Создайте процедуру обработки события *щёлкнуть кнопкой мыши.* Для этого выберите в таблице окно *Инспектора объектов* вкладки *События*, дважды щёлкните в ячейке справа от `OnMouseDown`. В окне редактора кода запишите программный код для объявления переменной логического типа `DownM`, которая будет передавать состояние нажатия мыши в каждую процедуру обработки события (рис. 27.12). Обратите внимание, что мы записываем её в программный код до описания процедуры обработки события!

Интересно

Переменная в программе, значения которой могут быть использованы и изменены во всех процедурах, составляющих программу, называется **глобальной переменной**.

```
var
    Form1: TForm1;
    DownM: Boolean;
implementation
```

Рис. 27.12

5. В программный код запишите процедуру обработки событий для формы `OnMouseDown` — нажата кнопка мыши, `OnMouseUp` — отпущена кнопка мыши, `OnMouseMove` — перемещение мыши (рис. 27.13).

```
procedure TForm1.FormMouseDown(Seder: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: integer);
begin
  DownM := True;
  Canvas.MoveTo(X,Y);

end;

procedure TForm1.FormMouseMove(Seder: TObject; Shift: TShiftState; X, Y: integer);
begin
  if DownM = True then Canvas.LineTo (X,Y);
end;

procedure TForm1.FormMouseUp(Seder: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: integer);
begin
  DownM := False;
end;
```

Рис. 27.13

6. Запустите проект на выполнение. Попробуйте нарисовать линиями своё имя.
7. Завершите работу с проектом и средой программирования.

4. Как построить изображение в среде программирования *PyCharm*?

В среде программирования *PyCharm* построение линий и других графических объектов в процессе выполнения программы аналогично описанному в среде *Lazarus*.

Для построения графических объектов вызывают соответствующий метод модуля `tkinter`, импортируемый в проект (рис. 27.14).

```
tkinter.Canvas(<название формы>, width=<значение>, height=<значение>)
```

Создание холста, размер которого задан значениями `width` — ширина, `height` — высота

Рис. 27.14

Создание изображений на холсте вызывается методом:

```
<Имя_объекта_холст>.create_<объект >.
```

Можно построить такие объекты:

- линия — `line(x1,y1,x2,y2)`
- прямоугольник — `rectangle(x1,y1,x2,y2)`
- ломаная линия — `polygon(x1,y1,x2,y2,x3,y3)`
- эллипс — `oval(x1,y1,x2,y2)`
- дуга — `arc(x1,y1,x2,y2,x3,y3,x4,y4)`

Объекты могут иметь параметры: `fill` — цвет заливки, `dash` — тип заливки.

Например, для рисования графических примитивов (рис. 27.15) в окне редактора кода записывают программный код (рис. 27.16).

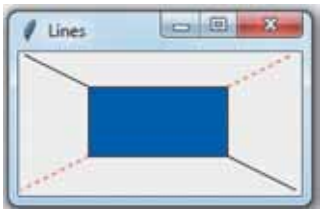


Рис. 27.15

```
import tkinter
main = tkinter.Tk()
main.wm_title("Lines")
# создание холста для рисования и размещения
на главной форме
cnv = tkinter.Canvas(main, width=200, height=100)
cnv.pack()
# команды рисования - создание графических примитивов
на холсте
cnv.create_line(0, 0, 200, 100)
cnv.create_line(0, 100, 200, 0, fill="red", dash=(4, 4))
cnv.create_rectangle(50, 25, 150, 75, fill="blue")
# запуск обработки событий программы
main.mainloop()
```

Рис. 27.16

ДЕЙСТВУЕМ

Упражнение 4. Столбчатая диаграмма.

Задание. В среде программирования *PyCharm* разработайте проект, в котором в окне главной формы будет построена столбчатая диаграмма, демонстрирующая соотношение между числами 15, 50, 70, 25, 10, 30.

1. Откройте среду *PyCharm*. В папке *Учебные проекты* своей структуры папок создайте проект *Диаграмма*, в котором создайте новую программу на языке *Python* с именем *Построение_диаграммы*.
2. В окне редактора кода запишите программу (рис. 27.17).

```
import tkinter
data = [15, 50, 70, 25, 10, 30]
main = tkinter.Tk()
main.wm_title("Data")
# создание холста для рисования и размещения
на главной форме
cnv = tkinter.Canvas(main, width=200, height=100)
cnv.pack()
# команды рисования -
создание графических примитивов на холсте
cnv.create_line(0, 100, 200, 100)
i = 0
while i < len(data):
 cnv.create_rectangle(10 + i * 30, 100, 30 + i * 30, 100 -
data[i], fill="blue")
 i += 1
# запуск обработки событий программы
main.mainloop()
```

Объявление списка значений

Ввод заголовка формы

Размеры холста

Построение 6 прямоугольников по координатам

Рис. 27.17

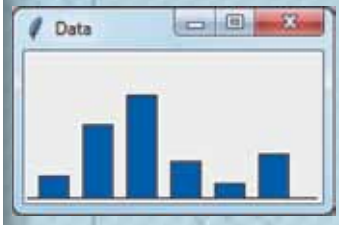


Рис. 27.18

3. Запустите проект на выполнение. Проверьте, получили ли вы окно формы, изображённое на рисунке 27.18.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 27* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Обсудите, как создать таблицу обобщения работы с графическими объектами в среде программирования *Lazarus*. Создайте её средствами текстового процессора.
2. Обсудите, чем похожи и чем отличаются рисование фигурами и построение этих фигур по точкам в среде программирования *Lazarus.* Постройте диаграмму Венна в среде текстового процессора.
3. Обсудите, как изменится проект *Линии*, если на форму разместить кнопку, нажатие которой вызывает событие заливки прямоугольника белым цветом:

```
Form1.Canvas.Brush.Color := clWhite;
Form1.Canvas.FillRect(0,0,500,300);
```

Какие элементы управления нужно использовать, чтобы в проекте можно было также задавать размер и цвет линии рисования? Доработайте проект в соответствии с вашими идеями.

4. Обсудите идею проекта, в котором можно продемонстрировать различные стили заливки фигур. Реализуйте этот проект в среде программирования *Lazarus*. Распределите роли: один разрабатывает графический интерфейс, другой — составляет программный код.
5. Обсудите идею проекта, в котором можно продемонстрировать построение линий, прямоугольников, эллипсов, ломаных, дуг и секторов. Реализуйте этот проект в средах программирования *Lazarus* и *PyCharm*. Распределите роли: один составляет проект в среде *Lazarus,* другой — в *PyCharm*.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Создайте проект *Полюса магнитов*, содержащий два рисунка магнитов, расположенных друг напротив друга. Необходимые изображения находятся в папке *Программирование* в файлах *Магнит1*, *Магнит2*. Если щёлкнуть мышью на левом рисунке, они оба приближаются друг к другу, а на правом — удаляются.
2. Создайте проект *Котёнок*, в котором котёнок после нажатия кнопки *Бежать* меняет своё положение. Различные виды котёнка получите из изображения *Котёнок1*, находящегося в папке *Программирование,* известными вам средствами обработки изображений (рис. 27.19).

Рис. 27.19

3. Создайте проект *Площадь фигур*, окно которого содержит изображения треугольника, прямоугольника и круга и обрабатываются следующие события: при наведении указателя мыши на каждую фигуру и удержании нажатой её левой кнопки на экранной форме в надписи выводится формула площади соответствующей фигуры; если кнопку отпустить, формула исчезает.

4. Разработайте проект *Прямоугольник*, в котором в текстовые поля или в список данных вводят пары координат, нажимают кнопку *Построить* и получают прямоугольник, построенный по указанным значениям координат.

5. Разработайте проект *Природные зоны*, в котором на форме дано изображение карты Украины, содержащееся в файле *Карта* в папке *Программирование*. После выбора в списке одной из природных зон равнинной части Украины: зона смешанных и широколиственных лесов, лесостепная зона, степная зона — пределы соответствующей зоны выделяются ломаной красного цвета.

6. Разработайте проект *Энциклопедия*, в котором на экранной форме *Сенсорная система* после нажатия кнопки *Отобразить* можно получить сведения об основных сенсорных системах: зрительной, слуховой, вкусовой, обонятельной, равновесия, движения, прикосновения, температуры, боли. Способ отображения информации — рисунок, текст — выбирается с помощью элементов управления. Выбор сенсорной системы — соответствующим переключателем. Необходимые сведения и изображения найдите в Интернете.

7. Разработайте проект *Лепестки*, в котором выбирают размер лепестка для рисования с помощью переключателя, а после нажатия кнопки *Рисовать* на экране строится изображение из 5 лепестков — секторов, координаты которых задаются случайным образом из диапазона $150 < x < 350$, $50 < y < 250$.

8. Разработайте игру *Найди пару*, в которой в таблице «спрятано» 6 пар ромбов, каждая из пар — одинакового цвета. При выборе пользователем определённой ячейки таблицы появляется соответствующий ромб. Если при следующем нажатии появляется ромб того же цвета, то обе фигуры «вылетают» из таблицы в правое поле. Если нет — они снова «прячутся» в таблицу (рис. 27.20).

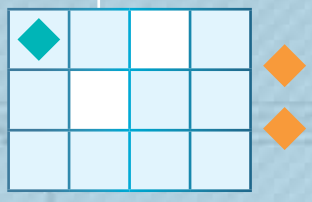


Рис. 27.20

ПОЛЕЗНЫЕ ССЫЛКИ

Курсы *Python* для начинающих:
http://foxford.ru/courses/199?ref=p191_pythonworldru
Сайт *Python 3* для начинающих:
http://pythonworld.ru/
Обучение языку *Python* в игровой форме:
https://checkio.org/
Курс по библиотеке *Tkinter* языка *Python*:
https://ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python

28. ПРАКТИЧЕСКАЯ РАБОТА 13

СОСТАВЛЕНИЕ И ВЫПОЛНЕНИЕ АЛГОРИТМОВ С ГРАФИЧЕСКИМ ОТОБРАЖЕНИЕМ ДАННЫХ

ВСПОМНИТЕ

- Как в среде программирования *Lazarus* используют рисунки с внешних носителей и создают изображения;
- как строят изображения в среде программирования *PyCharm*.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 13.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Движение (*24 балла*)

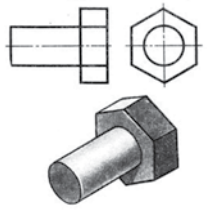
В среде программирования *Lazarus* разработайте проект *Движение*, в котором после нажатия кнопки *Поехать* запускается демонстрация движения автомобилей через перекрёсток без поворотов, регулируемый светофором. Свет светофора выбирается в группе переключателей *Светофор*, вид автомобиля — в группе флажков *Транспорт*, допустимая скорость на участке дороги — в группе переключателей *Скорость* (50, 70, 90 км/ч). Светофор и перекрёсток нарисуйте средствами среды программирования, а изображения автомобилей загрузите из файлов *Авто1, Авто2, Авто3*, находящихся в папке *Программирование.*

Задание 2. Графический редактор (*24 балла*)

В среде программирования *Lazarus* разработайте проект *Графический редактор*, в котором можно получать изображения штампов — кругов, ромбов, овалов, прямоугольников, квадратов, прямоугольников со скруглёнными краями, повёрнутых ромбов и квадратов, выбираемых с помощью флажков и строящихся после нажатия кнопки *Построить* по начальным координатам, введённым в текстовые поля. Количество изображений, повторяемых штампом по горизонтали через некоторый интервал, задаётся в программе случайно (количество от 1 до 5, интервал — от 10 до 50). Цвет рисования и стиль закрашивания выбирается в группах соответствующих переключателей.

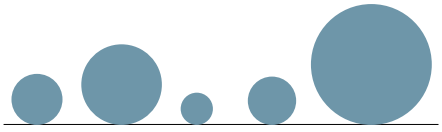
Задание 3. Чертёж детали (*20 баллов*)

Разработайте в среде *PyCharm* проект, с помощью которого на экране строится чертёж детали. Самостоятельно подберите координаты геометрических фигур, составляющих изображение проекций, соблюдая пропорции на рисунке.



Задание 4. Пузырьковая диаграмма (*22 балла*)

Разработайте проект в среде *PyCharm*, с помощью которого по введённым данным — списку пяти числовых значений — строится пузырьковая диаграмма, отображающая «пузырьки», размер которых соответствует числовым значениям.



РАЗДЕЛ 7

ТЕХНОЛОГИИ РАБОТЫ С ЧИСЛОВЫМИ ДАННЫМИ В ТАБЛИЧНОМ ПРОЦЕССОРЕ

29. ВЫЧИСЛЕНИЯ В ЭЛЕКТРОННЫХ ТАБЛИЦАХ

ВСПОМНИТЕ:

- с какими основными объектами можно работать в табличном процессоре;
- как применять средство автозаполнения для ввода данных;
- как копировать и перемещать данные из ячеек и диапазона ячеек;
- как изменять форматирование таблицы;
- как выполнять вычисления с числовыми данными таблицы;
- как выполнять вычисления с помощью встроенных функций;
- что происходит при копировании формул.

ВЫ УЗНАЕТЕ:

- чем отличаются абсолютные и относительные ссылки на ячейки;
- как добавить в формулу встроенную функцию;
- какие математические и статистические функции часто используются в табличном процессоре;
- как в формулах используются логические функции;
- как использовать условное форматирование данных.

ИЗУЧАЕМ

1. Чем отличаются абсолютные и относительные ссылки на ячейки?

Вы уже знаете, что вычисления в электронных таблицах выполняются с помощью формул. В формулах используют ссылки на ячейки или диапазоны ячеек, указывая их адреса либо имя ячейки или диапазона, заданные пользователем.

При работе с электронными таблицами однотипные расчёты часто выполняют в нескольких смежных ячейках. В табличном процессоре, в частности *Microsoft Excel 2010* или *LibreOffice Calc*, можно в этом случае не вводить формулы несколько раз с клавиатуры, а скопировать формулу из одной ячейки в другую с помощью буфера обмена или воспользоваться средством автозаполнения. При этом также происходит копирование формулы.

В отличие от копирования текстовых значений, при копировании формулы, содержащей ссылки на ячейки, можно получить формулу, не дублирующую исходную. Это зависит от вида ссылок на ячейки, которые могут быть **относительными, абсолютными** или **комбинированными** (**смешанными**). Вид ссылок важен только при копировании формулы, при вычислении значения по формуле в одной ячейке вид ссылок роли не играет.

Вы уже создавали формулы, содержащие относительные ссылки — такие ссылки используются по умолчанию. При копировании формулы, содержащей **относительную ссылку**, такая ссылка будет скорректирована в зависимости от направления копирования в электронной таблице. Относительная ссылка на ячейку состоит только из названия столбца и номера строки.

Например, если ячейка *F4* содержит формулу *=D4*E4*, то при копировании её в ячейку *F5* она примет вид *=D5*E5*. Такая формула с относительными

ссылками при копировании в новую ячейку интерпретируется так: найти произведение значений двух соседних слева ячеек в той же строке, что и ячейка, содержащая формулу.

Абсолютная ссылка в формуле показывает, что при копировании формулы необходимо оставлять ссылку именно на ту ячейку, адрес которой указан, такой адрес останется неизменным. Чтобы отличить абсолютную ссылку от относительной, в её записи перед названием столбца и номером строки ставится символ *$*, например, *$C$1* (рис. 29.1).

	A	B	C	D	E
1	Процент скидки		5%		
2					
3	**Наименование товара**	**Стоимость**	**К оплате**		
4	Наушники A4tech MK-610	109,3	103,84	=B4*(1-$C$1)	
5	Мышь TRUST GXT 25 Gaming Mouse	299	284,05	=B5*(1-$C$1)	
6	Салфетки ColorWay CW-1071	49	46,55	=B6*(1-$C$1)	
7					

Рис. 29.1

В **комбинированной, или смешанной, ссылке** название столбца является абсолютным, а номер строки — относительным, или наоборот. Например, *B$4*, *$B4* (рис. 29.2). Корректируется при копировании только относительная часть адреса.

	A	B	C	D	E	F	G	H	I
1	**Прыжки в длину**								
2			Попытка			Отклонение			
3	Спортсмен	Рекорд	1	2	3	1	2	3	=C4-$B4
4	Андриевский	466	460	468	463	-6	2	-3	
5	Васильковец	485	482	485	480	-3	0	-5	
6									
7									
8									

=C5-$B5 =E5-$B5

Рис. 29.2

Алгоритм выполнения вычислений в табличном процессоре с однотипными расчётами можно представить так:

1. Внести в ячейки таблицы данные, необходимые для выполнения вычислений.
2. Определить, какие ячейки в таблице должны содержать однотипные расчёты, какие ссылки следует использовать для вычислений — абсолютные, относительные или комбинированные.
3. Создать формулу со ссылками на ячейки, используя выделение в таблице нужных ячеек или диапазонов.
4. Указать при необходимости те ссылки, которые являются абсолютными или комбинированными, добавив знак *$* перед названием столбца и (или) номером строки.
5. Скопировать созданную формулу в диапазон ячеек, содержащих однотипные расчёты.

ДЕЙСТВУЕМ

Упражнение 1. Расчёты.

Задание. Вычислите стоимость проданного товара с помощью ввода формул для данных таблицы, содержащихся в файле *Расчёты* папки *Электронные таблицы*.

1. Создайте в своей структуре папок папку *Табличный процессор*.
2. В папке *Электронные таблицы* откройте файл *Расчёты*.
3. Выделите ячейку *D3*, введите в неё формулу *=C3*$C$12* (рис. 29.3) и нажмите клавишу *Enter*.
   Цена в гривнях рассчитывается как произведение цены в долларах на курс доллара, однако адрес ячейки *C12*, в которой введён курс доллара, не нужно менять при копировании формулы в другие ячейки. Поэтому формула содержит относительный адрес *C3*, корректируемый при копировании, и абсолютный адрес *$C$12*.
4. Выделите ячейку *D3* и выполните протягивание за маркер автозаполнения вниз до ячейки *D7*.

Интересно

В табличном процессоре *Microsoft Excel 2010* можно быстро изменить вид ссылки в формуле, для этого нужно нажать на клавиатуре клавишу *F4* — её последовательное нажатие меняет относительную ссылку на абсолютную, затем на комбинированную с абсолютным номером строки, далее — на комбинированную с абсолютным названием столбца и снова на относительную ссылку.

D3 =C3*$C$12

	A	B	C	D	E	F	G	H
1	Расчёт стоимости проданного товара							
2	№	Товар	Цена в дол.	Цена в грн	Цена с НДС	Количество	Стоимость	
3	1	Фоторамка	$6,00	141,24 грн.		3		
4	2	Портативная колонка	$10,60			2		
5	3	Электронная книга	$65,00			2		
6	4	Карта памяти 16 ГБ	$5,56			5		
7	5	Карта памяти 32 ГБ	$12,24			4		
8								
9								
10		Дата выписки счёта	12.01.2016					
11		Дата оплаты счёта	17.01.2016					
12		Курс дол.	23,54					
13								
14		Стоимость покупки						

Рис. 29.3

5. Для вычисления цены товара с НДС (налог на добавленную стоимость составляет 20 % от стоимости товара) необходимо к цене товара прибавить ещё 20 % его стоимости, поэтому в ячейку *E3* введите формулу = *D3+D3* 0,2* и нажмите клавишу *Enter.* Для дальнейшего копирования этой формулы в ячейки столбца *E* необходимо, чтобы адрес ячейки *D3* изменялся соответственно на *D4*, затем на *D5*, далее на *D6* и *D7*, — значит, нужно в формуле оставить относительный адрес ячейки *D3*.
6. С помощью автозаполнения скопируйте формулу из ячейки *E3* в ячейки диапазона *E4:E7*.
7. Выделите ячейку *G3* и введите формулу = *E3*F3* для вычисления стоимости указанного количества товара. Скопируйте эту формулу в диапазон ячеек *G4:G7*.
8. Выделите ячейку *G14*. С помощью автосуммы найдите сумму значений диапазона ячеек *G3:G7*.
9. Сохраните результаты в файле с тем же именем в папке *Табличный процессор* своей структуры папок.

Упражнение 2. Таблица квадратов.

Задание. Дополните таблицу в файле *Таблица квадратов* папки *Электронные таблицы*, чтобы получить таблицу квадратов двузначных чисел с использованием формул, содержащих комбинированные ссылки.

B3 =($A3*10+B$2)^2

	A	B	C	D	E	F	G	H	I	J	K
1	Таблица квадратов двузначных чисел										
2		0	1	2	3	4	5	6	7	8	9
3	1	100	121	144	169	196	225	256	289	324	361
4	2	400	441	484	529	576	625	676	729	784	841
5	3	900	961	1024	1089	1156	1225	1296	1369	1444	1521
6	4	1600	1681	1764	1849	1936	2025	2116	2209	2304	2401
7	5	2500	2601	2704	2809	2916	3025	3136	3249	3364	3481
8	6	3600	3721	3844	3969	4096	4225	4356	4489	4624	4761
9	7	4900	5041	5184	5329	5476	5625	5776	5929	6084	6241
10	8	6400	6561	6724	6889	7056	7225	7396	7569	7744	7921
11	9	8100	8281	8464	8649	8836	9025	9216	9409	9604	9801
12											

Рис. 29.4

1. В папке *Электронные таблицы* откройте файл *Таблица квадратов.*
2. Выделите ячейку *B3* и введите в неё формулу = *($A3*10+B$2)^2* (рис. 29.4).
3. Выделите ячейку с формулой и выполните протягивание за маркер автозаполнения вправо до ячейки *K3*. Не снимая выделения с диапазона, выполните протягивание за маркер автозаполнения вниз ячейки *K11*.
4. Сохраните результаты в файле с тем же именем в папке *Табличный процессор* своей структуры папок.

2. Как добавить в формулу встроенную функцию?

Вы уже умеете использовать для вычислений встроенные функции табличного процессора для определения суммы значений диапазона ячеек, среднего значения, максимального или минимального значений.

Табличный процессор содержит большой набор встроенных функций, которые можно использовать для вычислений. Каждая функция имеет своё имя, большинство функций содержит по крайней мере один необходимый для вычисления значения функции аргумент. Аргументы записываются в круглых скобках и отделяются друг от друга точкой с запятой (;). Аргументом может быть число, текст, записанный в кавычках, ссылки на ячейку или диапазон ячеек, либо выражение, содержащее функции. Некоторые аргументы являются обязательными, другие — нет.

Функцию можно ввести с клавиатуры, как и любое содержимое ячейки. Для упрощения ввода функций в формулу можно воспользоваться соответствующими инструментами табличного процессора. Для облегчения поиска все функции объединены в категории: математические, статистические, логические, финансовые, текстовые и т. д.

Также выделена отдельная категория *Недавно использовались в Microsoft Excel 2010* (*Последние использованные* в *LibreOffice Calc*), где можно просмотреть и выбрать функции, которые использовались в последнее время при работе на конкретном компьютере. Как правило, в этой категории будут отображены наиболее часто употребляемые функции, поэтому в ней можно быстро найти нужную функцию.

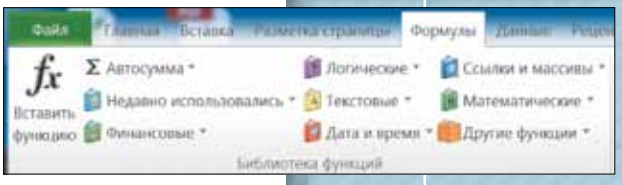


Рис. 29.5

В табличном процессоре *Microsoft Excel 2010* найти и добавить в формулу функцию можно с помощью инструментов из группы *Библиотека функций* на вкладке *Формулы* (рис. 29.5).

В табличных процессорах слева от строки формул расположен инструмент *Вставить функцию* —  или  , с помощью которого вызывают мастер функций. При этом открывается окно *Мастер функций — шаг 1 из 2* в *Microsoft Excel 2010* (рис. 29.6, *а*) или *Мастер функций* в *LibreOffice Calc* (рис. 29.6, *б*).

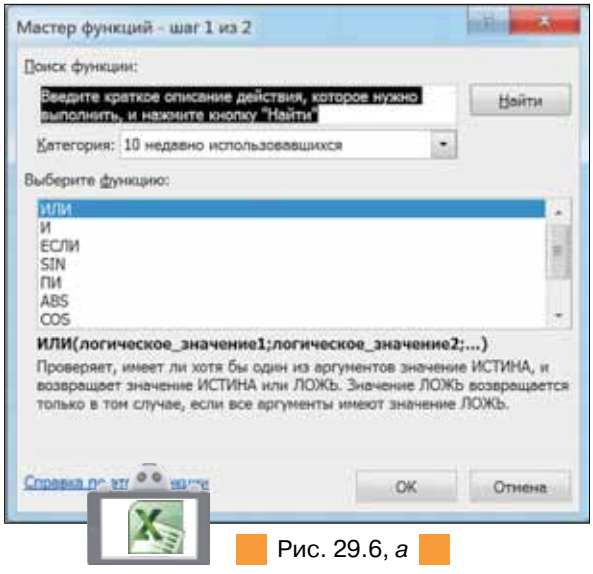


Рис. 29.6, *а*

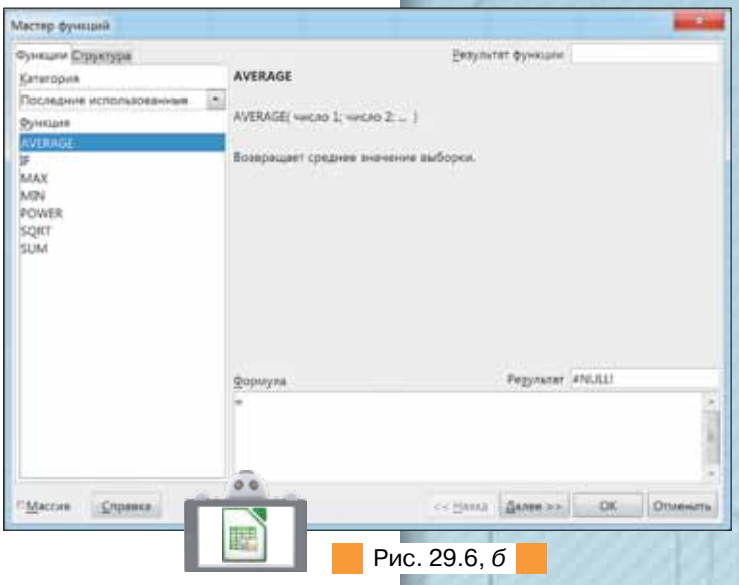


Рис. 29.6, *б*

Интересно

В табличном процессоре *Microsoft Excel* с русским интерфейсом названия большинства функций представлены сокращениями от русских слов.

Некоторые функции не имеют аргументов, но даже в этом случае круглые скобки всё равно должны быть записаны. Например, функция ПИ() возвращает значение числа π с точностью до 15 знаков.

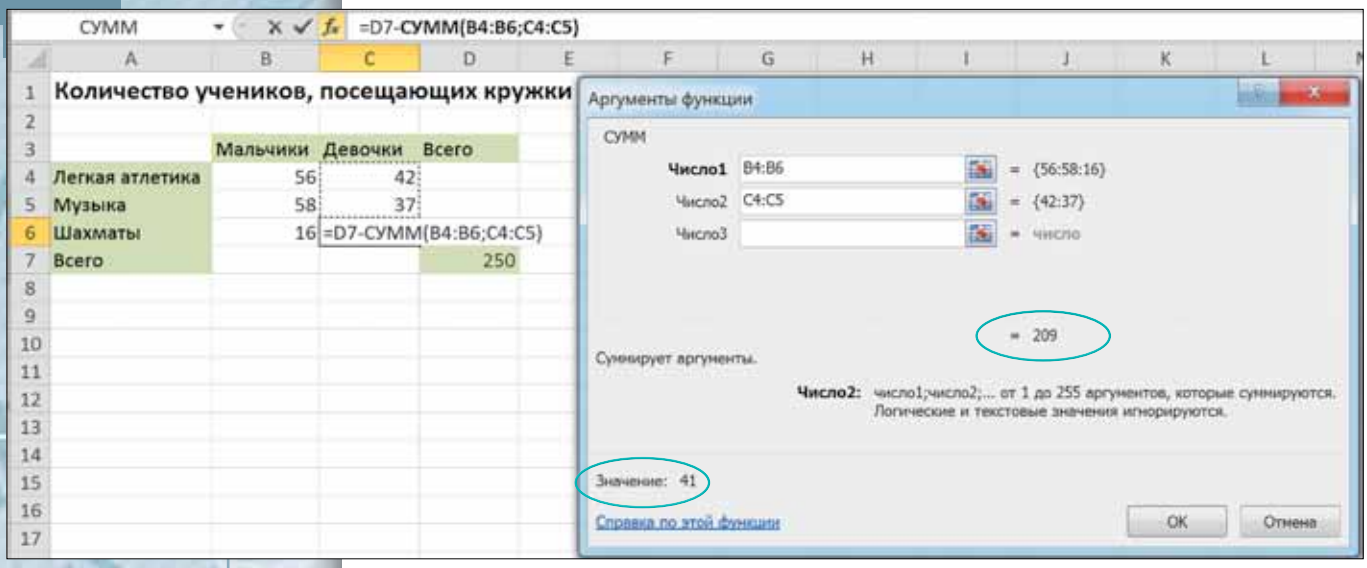


Рис. 29.7

В этом окне необходимо сначала выбрать категорию функции и имя нужной функции из списка. При необходимости можно воспользоваться справкой об использовании каждой функции. На следующем шаге следует задать значения аргументов функции. Это можно сделать, выделив ячейки в рабочем поле с помощью мыши или введя адреса ячеек с помощью клавиатуры. Аргументы функций, как и ссылки в формулах, могут содержать абсолютные, относительные или комбинированные ссылки — в зависимости от того, должны ли ссылки меняться при копировании формулы с функцией. После завершения ввода аргументов отображается значение функции, а в области *Значение* — результат вычисления формулы (рис. 29.7).

ДЕЙСТВУЕМ

Упражнение 3. Создание формулы с использованием функции для вычисления.

Задание. Ученикам 8 и 9 классов предложили выбрать один из кружков: лёгкой атлетики, музыки и шахмат. Среди 250 учеников 56 мальчиков и 42 девочки выбрали лёгкую атлетику, 58 мальчиков и 37 девочек — музыку, 16 мальчиков выбрали шахматы. В электронной таблице *Кружки*, находящейся в папке *Электронные таблицы*, создайте формулы для вычисления количества девочек, выбравших шахматы, и общего количества детей, выбравших каждый кружок.

1. В папке *Электронные таблицы* откройте файл *Кружки*.
2. Для определения количества девочек, выбравших шахматы, следует из общего количества всех учащихся вычесть количество всех мальчиков, а также девочек, выбравших другие кружки. Для создания такой формулы выделите ячейку *C6*, введите символ «=», щёлкните на ячейке *D7*, содержащей значения 250, и введите символ «–». Выберите инструмент *Вставить функцию*  слева от строки формул и просмотрите список функций в категории *Недавно использованные*. Если среди них есть функция СУММ, выберите её. Иначе выберите категорию *Математические* и выберите эту функцию.
3. Для аргумента *Число 1* выделите диапазон ячеек *B4:B6*, для аргумента *Число 2* — диапазон *C4:C5* (рис. 29.5). Нажмите кнопку *OK*.

4. Используя инструмент *Автосумма*, создайте в ячейке *D4* формулу для вычисления суммы значений для диапазона ячеек *B4:C4*.
5. С помощью автозаполнения скопируйте созданную формулу в ячейки *D5* и *D6*.
6. Сохраните результат в файле с тем же именем в папке *Табличный процессор* своей структуры папок.

3. Какие математические и статистические функции часто используются в табличном процессоре?

Наиболее часто используемые функции, которые можно добавить в формулу с помощью средства *Автосумма*, можно найти также в категориях *Математические* или *Статистические*. Например, функция СУММ (SUM) входит в категорию *Математические*. Примерами функций из категории *Статистические* являются СРЗНАЧ (AVERAGE) для определения среднего значения диапазона ячеек, СЧЁТ (COUNT) — количества непустых ячеек в заданном диапазоне, МАКС (MAX) и МИН (MIN) соответственно — для определения наибольшего и наименьшего значений.

Для решения задач по алгебре и геометрии можно использовать встроенные в табличный процессор математические функции. Наиболее употребляемые из них представлены в таблице 29.1.

Таблица 29.1

Функция (рус. интерфейс)	Функция (укр. и англ. интерфейс)	Результат
ABS(число)	ABS(число)	Модуль (абсолютное значение) аргумента
COS(число)	COS(число)	Косинус аргумента, заданного в радианах
ГРАДУСЫ(число)	DEGREES(число)	Преобразовывает значение угла, заданного в радианах, в градусы
ОСТАТ(число; делитель)	MOD(число; делитель)	Остаток от деления заданного числа на указанный делитель
SIN(число)	SIN(число)	Синус аргумента, заданного в радианах
TAN(число)	TAN(число)	Тангенс аргумента, заданного в радианах
ПИ()	PI()	Значение числа π с точностью до 15 знаков
СТЕПЕНЬ(число; степень)	POWER(число; степень)	Результат возведения числа в указанную степень
РАДИАНЫ(число)	RADIANS(число)	Преобразовывает значение угла, заданного в градусах, в радианы
ОКРУГЛ(число; количество знаков)	ROUND(число; количество знаков)	Число, округлённое до указанного количества знаков после запятой
КОРЕНЬ(число)	SQRT(число)	Значение арифметического квадратного корня аргумента

ДЕЙСТВУЕМ

Упражнение 4. Использование математических функций в формулах.
Задание. Для прямоугольного треугольника *ABC* вычислите длину гипотенузы и второго катета по значениям катета и противоположного ему угла, заданным в файле *Прямоугольный треугольник*, находящемся в папке *Электронные таблицы*.

1. В папке *Электронные таблицы* откройте файл *Прямоугольный треугольник*.
2. Выделите ячейку *B5*, на вкладке *Формулы* выберите категорию *Математические* и функцию РАДИАНЫ, для ввода аргумента щёлкните на ячейке *B4*. Формула в ячейке *B5* примет вид = РАДИАНЫ (*B4*).
3. Для вычисления длины гипотенузы следует найти частное длины катета и синуса противоположного угла. Для создания формулы выделите ячейку *B8*, введите символ «=», щёлкните на ячейке *B3* и введите символ «/». Выберите инструмент *Вставить функцию* f_x, в категории *Математические* найдите и выберите функцию SIN. Обратите внимание, что аргументом этой функции должно быть значение в радианах, поэтому в поле *Число* следует ввести ссылку на ячейку *B5*. Формула примет вид =*B3*/SIN(*B5*).

Примечание. Формула может содержать несколько функций, например, если не вычислять значение угла *B* в радианах в ячейке *B5*, можно записать формулу в ячейке *B8* так: =*B3*/SIN(РАДИАНЫ(*B4*)).

4. Аналогично в ячейке *B9* создайте формулу для вычисления длины второго катета.
5. Для ячейки *B9* установите числовой формат с округлением до десятых (рис. 29.8).
6. Сохраните результат в файле с тем же именем в папке *Табличный процессор* своей структуры папок.

	A	B	C	D
1	В прямоугольном треугольнике ABC найди стороны AB и BC			
2				
3	Катет AC, см	4		
4	Угол B, в градусах	30		
5	Угол B, в радианах	0,523599		
6				
7				
8	Гипотенуза BC, см	8		
9	Катет AB, см	6,9		
10				

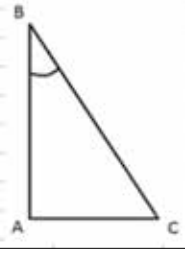


Рис. 29.8

4. Как в формулах используются логические функции?

В табличном процессоре используются также формулы, в которых аргументом является логическое выражение. Логическое выражение содержит знак сравнения и может принимать одно из двух значений: ИСТИНА (TRUE) или ЛОЖЬ (FALSE) в зависимости от значений в ячейках, на адреса которых есть ссылка в выражении. Примеры логических выражений представлены в таблице 29.2.

Таблица 29.2

Логическое выражение	Пояснение условия, при котором логическое выражение принимает значение ИСТИНА (TRUE)
A1>1	Числовое значение в ячейке *A1* больше 1
*F2***A4*=СУММ(*B2*:*B13*)	Произведение значений в ячейках *F2* и *A4* равно сумме значений диапазона *B2*:*B13*
B13=«Петя»	Содержимым ячейки *B13* является текстовое значение «Петя»
B3–*C3*>=12	Разность значений в ячейках *B3* и *C3* больше или равна 12

Если результат вычисления по формуле зависит от выполнения некоторого условия — истинности логического выражения, то используются **логические функции**. При составлении алгоритмов в этом случае вы используете алгорит-

мическую структуру ветвления. К категории логических функций относится функция ЕСЛИ (IF), с помощью которой можно организовать ветвление в табличном процессоре (рис. 29.9).

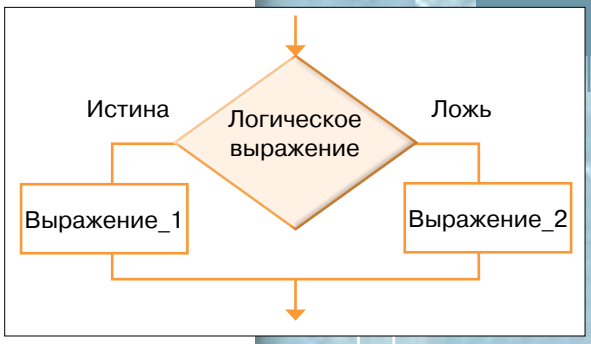


Рис. 29.9

Функция ЕСЛИ имеет три аргумента:
ЕСЛИ(*логическое_выражение; значение_если_истина; значение_если_ложь*).

При использовании функции ЕСЛИ проверяется истинность указанного логического выражения, и в ту ячейку, где находится формула, заносится *значение_если_истина*, если логическое выражение истинно, или *значение_если_ложь*, если логическое выражение ложно. Аргументами *значение_если_истина* и *значение_если_ложь* могут быть число, текстовое значение, записанное в кавычках, или выражение для вычисления. Выражение для вычисления может содержать другие функции, в том числе и вложенную функцию ЕСЛИ, если нужно сформировать более сложное условие.

Примеры формул, содержащих логическую функцию ЕСЛИ, представлены в таблице 29.3.

Таблица 29.3

Формула	Значения в ячейках	Результат вычисления по формуле
= ЕСЛИ(*A1***B2*>0; КОРЕНЬ(*A1***B2*); СТЕПЕНЬ(*A1*,*B2*))	*A1* = 2, *B2* = 18	6
	A1 = 5, *B2* = –1	0,2
	A1 = –1, *B2* = 4	1
= ЕСЛИ(*B2*>*C2*;”Превышение бюджета”; ”ОК”)	*B2* = 450, *C2* = 500	ОК
	B2 = 300, *C2* = 250	Превышение бюджета
= ЕСЛИ(*A10*=100;СУММ(*B5*:*B7*);””)	*A10* = 100, *B5* = 32, *B6* = 45, *B7* = 18	95
	A10 = 50, *B5* = 100, *B6* = 210, *B7* = 180	Пустая ячейка
= ЕСЛИ(*A1*>*A2*;100;0)	*A1* = 20, *A2* = 25	0
	A1 = 20, *A2* = 15	100

При использовании мастера функций каждый из аргументов функции ЕСЛИ записывают в отдельном поле (рис. 29.10, *а*, *б*).

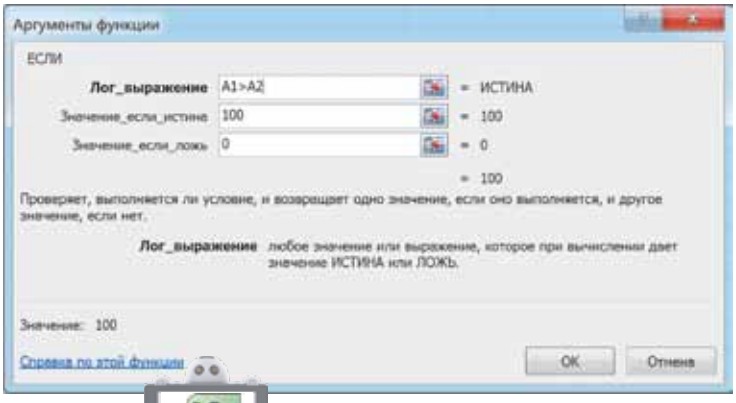


Рис. 29.10, *а*

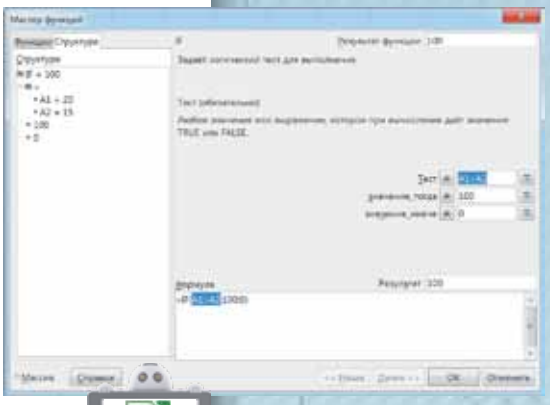


Рис. 29.10, *б*

Для создания составного логического выражения используют другие функции из категории *Логические*: И(*логзнач1; логзнач2 ...*) (AND(*логзнач1; логзнач2 ...*)), ИЛИ(*логзнач1; логзнач2 ...*) (OR(*логзнач1; логзнач2 ...*)), НЕ(*логзнач*) (NOT(*логзнач*)). Аргументами логических функций являются логические выражения, которые могут принимать одно из двух значений — истина или ложь. Их использование аналогично соответствующим командам, используемым для записи составных условий при составлении алгоритмов — программ.

Чаще всего логические функции И, ИЛИ, НЕ используют для записи составного условия в качестве аргумента логической функции ЕСЛИ.

ДЕЙСТВУЕМ

Упражнение 5. Использование логической функции ЕСЛИ.

Задание. Для данных роста учеников 8 класса, заданных в столбце *B* электронной таблицы *Рост*, в столбце *C* создайте формулу для определения цвета маркировки школьных парт при условии, что для учащихся, рост которых больше 160 см, маркировка должна быть зелёной, для остальных — красной.

1. В папке *Электронные таблицы* откройте файл *Рост*.
2. Выделите ячейку *C2*. Выберите инструмент *Вставить функцию* f_x, в категории *Логические* найдите и выберите функцию ЕСЛИ.
3. Выделите аргумент *Логическое выражение*, щёлкните на ячейке *B2*, введите с клавиатуры <=160.
4. Выделите аргумент *Значение_если_истина*, введите текст *Красная*. Обратите внимание, что текстовое значение автоматически заключается в кавычки.

	A	B	C
1	**Фамилия ученика**	**Рост**	**Маркировка**
2	Антоненко	145	Красная
3	Батайло	160	Красная
4	Овчар	162	Зелёная
5	Гнатишин	149	Красная
6	Эдгар	170	Зелёная

Рис. 29.11

5. Выделите аргумент *Значение_если_ложь*, введите текст *Зелёная*. Нажмите кнопку *ОК*. Убедитесь, что формула имеет вид: = ЕСЛИ(B2<= 160;"Красная";"Зелёная")
6. Выделите ячейку *C2*, воспользуйтесь автозаполнением для копирования формулы в диапазон ячеек *C3:C6*. Проверьте результат в диапазоне *C3:C6* (рис. 29.11).
7. Сохраните результаты работы в файле с тем же именем в папке *Табличный процессор* своей структуры папок.

5. Как использовать условное форматирование данных?

Изменить в электронной таблице форматирование данных, соответствующих определённым условиям, можно с помощью **условного форматирования.** Это средство используют для создания правил форматирования отдельных ячеек таблицы в зависимости от их значения.

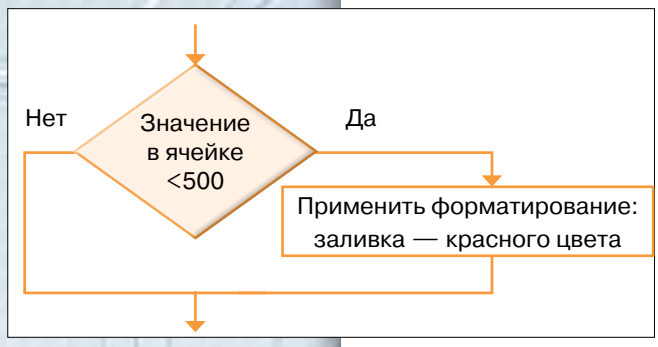


Рис. 29.12

Например, можно применить условное форматирование к ячейкам так, чтобы любое числовое значение, меньшее 500, отображалось в соответствующих ячейках на красном фоне. Условное форматирование соответствует команде ветвления (рис. 29.12).

Если значение в ячейке, которая входит в выделенный диапазон, < 500,

то применить к ячейке форматирование: заливка — красного цвета.

	A	B	C	D	E	F	G	H
1	**Учебные достижения учащихся**							
2								
3	*Фамилия*	*Алгебра*	*Геометрия*	*Физика*	*Информатика*	*Укр. лит.*	*Заруб. лит.*	*Биология*
4	Бондарь В.	11	11	10	12	10	11	11
5	Гапон С.	9	8	9	8	9	10	10
6	Стецюк К.	8	9	9	9	8	9	10
7	Савицкая О.	4	4	3	4	4	3	5
8	Ткаченко В.	9	9	10	11	11	11	10

Рис. 29.13

Табличные процессоры также позволяют применять условное форматирование с использованием гистограмм, цветовых шкал и наборов пиктограмм. Например, с помощью цветовой шкалы все наименьшие значения выделенного диапазона могут иметь красный цвет заливки, наибольшие значения — зелёный, а все промежуточные значения — другие оттенки от красного до зелёного (рис. 29.13).

Условное форматирование с использованием гистограмм, цветовых шкал и набора пиктограмм соответствует команде ветвления с вложениями. Например, условное форматирование с использованием цветовой шкалы по шаблону  , результат которого отображён на рисунке 29.13, соответствует команде ветвления.

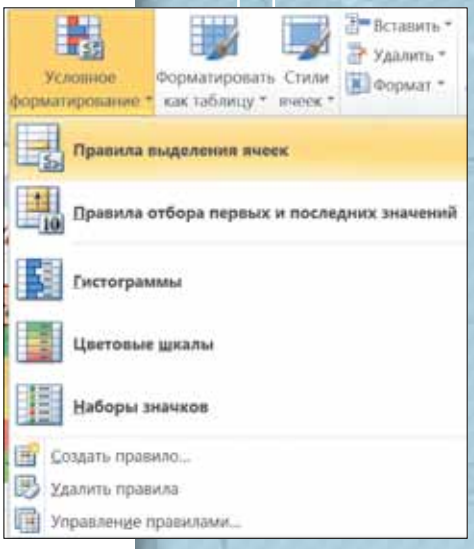


Рис. 29.14, *а*

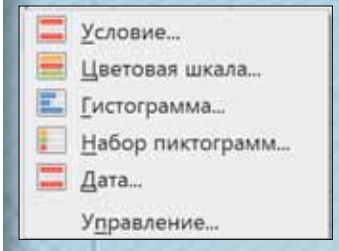


Рис. 29.14 *б*

Если значение в ячейке совпадает с **минимальным** значением диапазона,
то применить к ячейке форматирование: заливка — красного цвета,
иначе если значение в ячейке совпадает с **максимальным** значением диапазона,
то применить к ячейке форматирование: заливка — зелёного цвета,
иначе применить форматирование: заливка — оттенки от красного до зелёного.

Для условного форматирования содержимого ячеек необходимо выделить ячейки, на которые распространяется такое форматирование, на вкладке *Главная* в группе *Стили* выбрать инструмент *Условное форматирование* (для *Microsoft Excel 2010*) или выбрать в меню *Формат* команду *Условное форматирование* (для *LibreOffice Calc*). После этого следует выбрать тип условного форматирования для создания нового правила выделения ячеек или гистограммы, цветовые шкалы или наборы пиктограмм (рис. 29.14, *а*, *б*).

При создании нового правила выделения ячеек можно выбрать условие для форматирования, задать значения, с которыми следует сравнивать значения из диапазона ячеек, и указать необходимый для применения формат (рис. 29.15).

Если среди предложенных форматов ни один не подходит, можно выбрать вариант *Пользовательский*

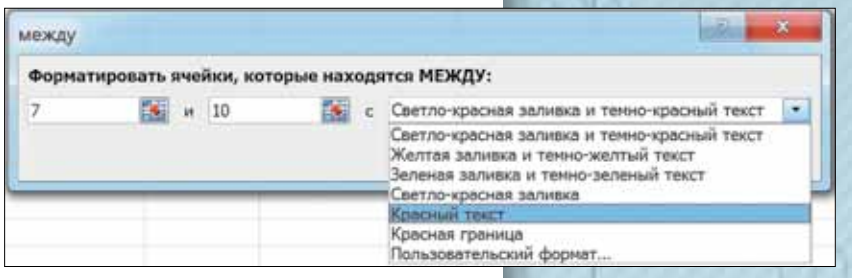


Рис. 29.15

формат, при этом будет отображено окно *Формат ячейки*, в котором можно задать параметры форматирования символов, заливки, границ и т. п.

Чтобы отменить условное форматирование в *Microsoft Excel 2010*, достаточно выделить диапазон с условным форматированием и выбрать команду *Удалить правила* из списка команд инструмента *Условное форматирование* (рис. 29.14, *а*). В табличном процессоре *LibreOffice Calc* для этого после выделения диапазона с условным форматированием следует выбрать команду *Формат/Условное форматирование/Управление* и в окне *Условное форматирование* выбрать кнопки *Удалить* и *ОК*.

ДЕЙСТВУЕМ

Упражнение 6. Компьютерная вышивка.

Задание. Примените условное форматирование к таблице *Вышивка* так, чтобы данные были отображены с использованием цветовой шкалы: наименьшие значения — на красном фоне, наибольшие — на зелёном, все промежуточные — в оттенках от красного до зелёного.

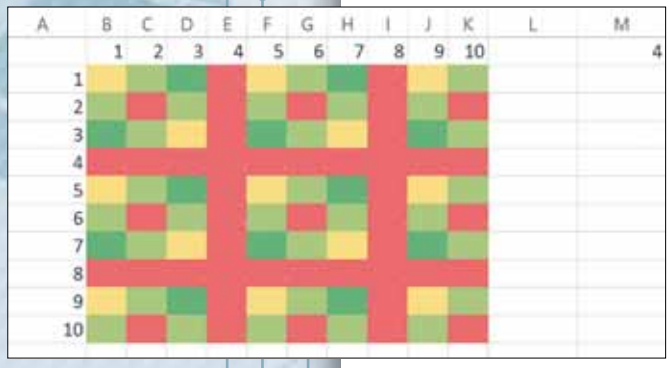


Рис. 29.16

1. В папке *Электронные таблицы* откройте файл *Вышивка*.
2. Выделите диапазон ячеек *B2:K11*, содержащий остаток от деления произведения номера строки и номера столбца на некоторое число, записанное в ячейке *M1*.
3. Выберите инструмент *Условное форматирование/Цветовые шкалы* на вкладке *Главная* (команда *Формат/Условное форматирование/Цветовая шкала*).
4. Среди шаблонов цветовых шкал выберите такой, чтобы наименьшее значение отображалось красным, наибольшее — зелёным, среднее — жёлтым (рис. 29.16).
5. Несколько раз измените значение в ячейке *M1*, чтобы получить наиболее понравившийся узор.
6. Сохраните результаты работы в файле с тем же именем в папке *Табличный процессор* своей структуры папок.

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 29* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Обсудите различия между абсолютными и относительными ссылками на ячейки электронной таблицы. Имеет ли комбинированная ссылка общие признаки абсолютной и относительной ссылок?
2. Поиграйте в игру *Аргументы функции*. Один участник называет имя функции из категорий *Математические*, *Статистические* или *Логические*, а другой приводит примеры её аргументов. Поменяйтесь ролями.

3. Ячейка *B2* содержит значение 20, ячейка *C2* — 10. В ячейку *D2* введена формула с использованием логической функции ЕСЛИ. По очереди определяйте, какое значение будет отображено в ячейке *D2,* и обоснуйте свой ответ.

1) =ЕСЛИ(*B2*<0; –*B2*; *B2*)
2) =ЕСЛИ(*B2*>*C2*; ”больше”; ””)
3) =ЕСЛИ(*C2*>0; *B2*+*C2*; *B2*–*C2*)

4. В ячейку *A1* введено число 100, а в ячейку *B1* — число 10. По очереди называйте значения логических функций и обоснуйте свой ответ:

1) И(*A1*<0; *B1*>0)
2) ИЛИ(*A1*<0; *B1*>0)
3) И(*A1*>20; *B1*<=50)
4) НЕ(*A1*<>0)
5) И(*A1*>50; *A1*<>70; *B1*=10)
6) ИЛИ(*A1*<200; *B1*<>0; *B1*<20)

РАБОТЕМ САМОСТОЯТЕЛЬНО

1. Выполните вычисления в файле электронной таблицы *Правило*, находящейся в папке *Электронные таблицы* (рис. 29.17). С помощью табличного процессора проверьте истинность утверждения $a \cdot b = НСК(a; b) \cdot НСД(a; b)$.

	A	B	C	D	E	F
1	Первое число	Второе число	Произведение	Наименьшее общее кратное	Наибольший общий делитель	Произведение НОК и НОД
2	a	b	ab	НОК(a,b)	НОД(a,b)	
3	48	12				

Рис. 29.17

2. В табличном процессоре составьте модель проверки условия всплывания тела. Учтите: если сила притяжения больше архимедовой силы, тело будет тонуть; если сила притяжения равна силе Архимеда, то тело может находиться в равновесии в любой точке жидкости; если сила притяжения меньше архимедовой силы, тело будет всплывать, подниматься вверх. Как вычислять силу Архимеда и силу притяжения, вспомните из уроков физики или найдите в Интернете.

	A	B	C	D	E	F	G
1	Отправление	Маршрут	Прибытие	Цена	Количество проданных билетов		Сумма
2	05:00	ТЕРНОПОЛЬ АВ	07:20	37,91	27		1023,57
3	04.01.16	- ЖАБЬНЯ					
4	05:10	ТЕРНОПОЛЬ АВ	07:40	50,81	19		965,39
5	04.01.16	- МОНАСТЫРИСКА					
6	05:15	ТЕРНОПОЛЬ АВ	08:45	93,66	25		2341,5
7	04.01.16	- РОВНО					
8	05:15	ТЕРНОПОЛЬ АВ	08:05	88,93	27		2401,11
9	04.01.16	- ИВ.ФРАНКОВСК 2					
10	05:20	КИЕВ АВ	10:30	100,45	25		2511,25
11	04.01.16	- ЯРЕМЧА					

Рис. 29.18

3. В файле *Касса* папки *Электронные таблицы* находятся данные о продаже билетов кассами автовокзала. Найдите стоимость билетов по каждому направлению и в целом. Примените условное форматирование *светло-красная заливка* и *тёмно-красный текст* для обозначения первых пяти значений с наибольшим количеством проданных билетов и в виде гистограммы для суммы от продажи билетов (рис. 29.18).

	A	B	C	D
1	Города	Расстояние до Киева, км	Количество необходимого топлива	Стоимость, грн
2	Львов			
3	Одесса			
4	Хмельницкий			
5	Винница			
6	Луцк			
7	Умань			
8				
9				

Рис. 29.19

4. В папке *Электронные таблицы* откройте файл *Квадратный корень*. Создайте формулу для вычисления значений арифметического квадратного корня для неотрицательных чисел. Для отрицательных значений предусмотрите текстовый комментарий «*Не существует*». Воспользуйтесь логической функцией ЕСЛИ и функцией КОРЕНЬ из категории *Математические*.

5. Создайте электронную таблицу для расчёта стоимости поездки автомобилем *Nissan Note* из Киева в города, указанные на рисунке 29.19. Расстоя-

ние можно узнать из Интернета или рассчитать по карте автомобильных дорог Украины, учитывая её масштаб. Стоимость 1 л топлива и расхода топлива на 100 км для данного автомобиля найдите в Интернете.

ИССЛЕДУЕМ

1. Определите, какие наборы пиктограмм можно использовать для условного форматирования данных в таблице. Для этого введите в таблицу значения 10, 20, 30, 40, выделите диапазон ячеек с этими данными, выберите инструмент *Условное форматирование* на вкладке *Главная* и *Наборы значков* (для *Microsoft Excel 2010*) или команду *Формат/Условное форматирование/Набор пиктограмм* (для *LibreOffice Calc*). Выбирайте различные наборы символов и определите, как меняется отображение данных для каждого из наборов (рис. 29.20).
2. Определите, сколько условий можно использовать одновременно при использовании условного форматирования.
3. Определите, правила каких типов можно создавать для условного форматирования данных таблицы.

Рис. 29.20

30. ПРАКТИЧЕСКАЯ РАБОТА 14

РЕШЕНИЕ ЗАДАЧ НА ВЫЧИСЛЕНИЕ

ВСПОМНИТЕ

- Как записывать абсолютные, относительные и комбинированные ссылки в формулах;
- как использовать автозаполнение для копирования формул;
- как применять встроенные функции для работы с данными.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 14.*

ПОМНИТЕ

При выполнении практических заданий соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Степени натуральных чисел (*8 баллов*)

Используя средства табличного процессора, создайте электронную таблицу степеней натуральных чисел первого десятка от первой степени до пятой.

Задание 2. Объём газа (*8 баллов*)

Используя средства табличного процессора, создайте электронную таблицу для определения объёма газа при нормальных условиях.

	A	B	C	D	E	F
1	Молярный объём газа при нормальных условиях, л/моль					22,4
2	**Газ**	**Масса газа, г**	**Молярная масса, г/моль**	**Объём, л**		
3	Кислород	0,6	32			
4	Водород	0,4	2			
5	Углекислый газ	1,8	44			

=B3/C3*$F$1

Задание 3. Энергопотребление (*10 баллов*)

Используя средства табличного процессора, создайте электронную таблицу для планирования экономии потребления электрической энергии. Воспользуйтесь файлом *Энергопотребление* для получения сведений о мощности бытовых устройств. Вычислите объём потреблённой электроэнергии за неделю в вашей семье и её стоимость. Спланируйте, где в расчётах должны быть абсолютные, а где — относительные ссылки.

Задание 4. Поездка (*11 баллов*)

Создайте средствами табличного процессора электронную таблицу, структура и формат которой изображены на рисунке, и сохраните её с именем *Поездка* в папке *Практическая работа 14*. В ячейки *C9:C12* введите данные и формулы в соответствии с заданием, записанным в ячейке *A4* таблицы.

	A	B	C
1	Дорога на отдых		
2			
3	Семья собирается в отпуск.		
4	Вычислите, какую сумму необходимо потратить на билеты при условии, что:		
5	1. Едут 2 взрослых и 1 ребёнок.		
6	2. Стоимость взрослого билета — 130 у.е.		
7	3. Детям — скидка 40 %.		
8	4. 1 у.е. = 23,54 грн		
9		Полный билет	130
10		Детский билет	78
11		*Всего в у.е.*	338
12		*Всего в грн*	7956,52

Задание 5. Закон Кулона (*11 баллов*)

Используя средства табличного процессора, создайте структуру расчётной таблицы для лабораторной работы по физике для определения электростатической силы взаимодействия двух заряженных частиц по закону Кулона по образцу и заполните её. Учтите, что закон Кулона имеет вид: $F_{12} = k \cdot \frac{q_1 \cdot q_2}{r_{12}^1}$.

Не забудьте использовать ссылки на ячейки при вводе расчётных формул.

	A	B	C	D	E
1	Закон Кулона				
2	Заряд1	Заряд 2	Расстояние между зарядами	Электростатическая сила взаимодействия	Электростатическая постоянная
3	q1(Кл)	q2(Кл)	r(м)	F	k(10^9 Н·м2·Кл-2)
4	0,0002	0,0000016	0,01	28760,7758	8,987742438
5	0,000008	0,0007	0,5	201,3254306	
6	0,00074	0,00008	0,001	532074352,3	

31. ПРАКТИЧЕСКАЯ РАБОТА 15

ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКИХ, ЛОГИЧЕСКИХ И СТАТИСТИЧЕСКИХ ФУНКЦИЙ ТАБЛИЧНОГО ПРОЦЕССОРА. УСЛОВНОЕ ФОРМАТИРОВАНИЕ

ВСПОМНИТЕ

- Как записывать абсолютные, относительные и комбинированные ссылки в формулах;
- как использовать автозаполнение для копирования формул;
- как применять математические, логические и статистические функции для работы с данными;
- как применять условное форматирование к ячейкам таблицы.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 15.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Часовые пояса (*8 баллов*)

Создайте электронную таблицу, с помощью которой можно определить время в Сиднее, Пекине, Кейптауне, Париже, Нью-Йорке, если задано время в Киеве. Воспользуйтесь картой часовых поясов. Примените к данным таблицы условное форматирование с цветовой шкалой.

Задание 2. Тематическая аттестация (*6 баллов*)

В папке *Электронные таблицы* откройте файл *Тематическая аттестация*.
Для данных этой таблицы найдите средний балл каждого ученика за текущие оценки по истории и итоговую оценку за тему.
Примените условное форматирование к соответствующему диапазону ячеек так, чтобы высокие результаты (10–12 баллов) отображались на зелёном фоне, достаточные (7–9 баллов) — на жёлтом, остальные — на розовом.

Задание 3. Таблица значений (*10 баллов*)

Создайте таблицу, содержащую значения синуса, косинуса и тангенса углов 0°, 10°, 20°, 30°, 40°, 50°, 60°, 70°, 80°, 90°. Обратите внимание, что аргументами соответствующих математических функций в табличном процессоре является значения угла в радианах.

Задание 4. Состав числа (*10 баллов*)

Используя математические функции вычисления остатка от деления и отбрасывания дробной части числа, создайте в табличном процессоре таблицу для определения цифр заданного трёхзначного числа, как показано на примере.

	100	10	1
189	1	8	9

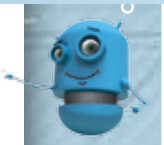
32. ДИАГРАММЫ РАЗНЫХ ТИПОВ. ПЕЧАТЬ ЭЛЕКТРОННОЙ ТАБЛИЦЫ

ВСПОМНИТЕ:

- из каких объектов состоит диаграмма;
- как создать столбчатую или круговую диаграмму в табличном процессоре;
- как настроить свойства объектов диаграммы.

ВЫ УЗНАЕТЕ:

- как «читать» диаграмму, построенную на основе данных таблицы;
- какой тип диаграммы выбрать;
- как изменять свойства отдельных составляющих диаграммы;
- как настроить значения параметров страницы перед печатью электронной таблицы;
- как напечатать электронную таблицу.



ИЗУЧАЕМ

1. Как «читать» диаграмму, построенную на основе данных таблицы?

Вы уже умеете создавать столбчатые и круговые диаграммы в табличном процессоре.

После построения диаграммы нужно правильно интерпретировать её, т. е. «читать» диаграмму и объяснять, что на ней изображено.

Чтобы диаграмму правильно интерпретировать, она должна содержать все необходимые составляющие: название диаграммы, подписи на осях, легенду и подписи данных (рис. 32.1). Если некоторые из этих объектов отсутствуют, диаграмму будет сложно анализировать.

Рис. 32.1

	A	B	C	D	E	F
1	Продажа туристических путевок					
2						
3	№	Страна	Город	апрель	май	июнь
4	1	Украина	Одесса	12	45	120
5	2	Украина	Херсон	3	10	35
6	3	Украина	Бердянск	4	13	60
7	4	Египет	Хургада	112	135	68
8	5	Египет	Шарм-эль-Шейх	126	145	80
9	6	Турция	Анталия	68	167	210
10	7	Турция	Кемер	45	180	230
11			Всего:	370	695	803

Источник данных

Рис. 32.2

Круговая диаграмма может быть построена только для одного ряда данных — когда данные являются составной частью одного целого; с помощью других типов, как правило, можно отображать несколько рядов данных.

Для таблицы в файле *Путёвки* (рис. 32.2) с помощью столбчатой диаграммы (рис. 32.1) можно сравнивать спрос на туры в конкретные города по количеству проданных путёвок за три месяца или сравнивать количество проданных путёвок в каждом месяце в конкретном направлении (город).

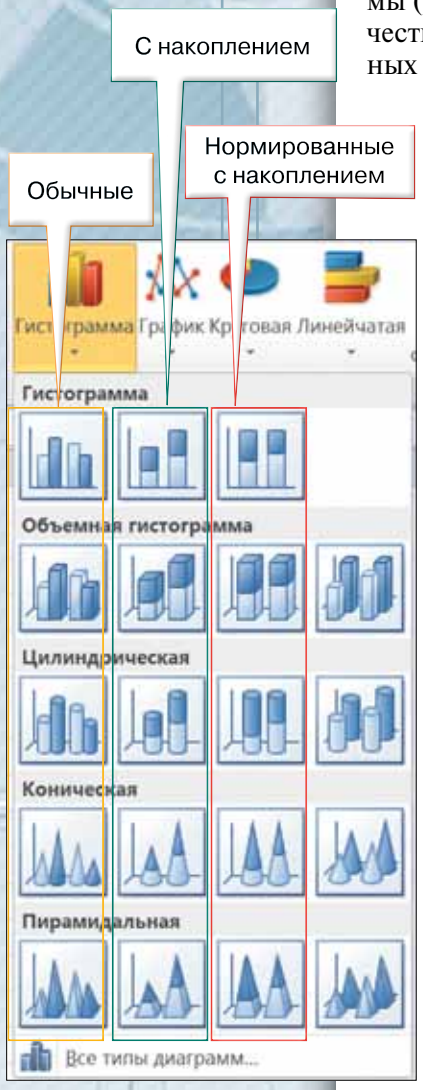


Рис. 32.3

ДЕЙСТВУЕМ

Упражнение 1. Анализ данных, отображённых на диаграмме.

Задание. Проанализируйте данные, отображённые на диаграмме, содержащейся в файле *Путёвки*, и объясните их.

По данным диаграммы, изображённой на рисунке 32.1, можно сделать следующие выводы:

- в начале туристического сезона, в апреле, наибольшее количество путёвок было продано в г. Хургада и г. Шарм-эль-Шейх (Египет), а наименьшее — в г. Херсон и г. Бердянск (Украина);
- в июне ситуация несколько изменилась — больше всего было продано путёвок в г. Кемер и г. Анталия (Турция), меньше всего — также в г. Бердянск (Украина);
- путёвки в г. Кемер становятся популярнее постепенно — с апреля (45) по июнь (230);
- наоборот, путёвки в г. Хургада больше продаются в апреле (112), чем в июне (68);
- ни в одном из месяцев, которые анализировались, не удалось продать больше 230 путевок в один город, наименьшее количество проданных путёвок за все месяцы — 3.

2. Какой тип диаграммы выбрать?

С помощью табличного процессора можно строить диаграммы различных типов.

Каждый тип содержит несколько видов диаграмм — плоские и объёмные, обычные, с накоплением или нормированные с накоплением. Например, гистограмма может быть одного из видов, изображённых на рисунке 32.3.

Прежде чем выбрать тип диаграммы, следует выделить **источник данных** — диапазон ячеек таблицы, содержащих данные, на основе

Рис. 32.4

которых будет создана диаграмма. Для построения диаграммы источник данных должен содержать числовые данные.

Названия типов диаграмм в различных табличных процессорах могут несколько отличаться — представляться синонимами, но они используются аналогично.

В табличном процессоре *Microsoft Excel 2010* выбрать тип диаграммы для выделенных данных можно на вкладке *Вставка* в группах *Диаграммы* и *Спарклайны* (рис. 32.4). Некоторые дополнительные типы диаграмм собраны в списке, который можно открыть с помощью инструмента *Другие*.

Если выбрать команду *Все диаграммы*, то можно увидеть список доступных типов и соответствующих видов диаграмм, отображённых в правой части окна *Вставка диаграммы* (рис. 32.5).

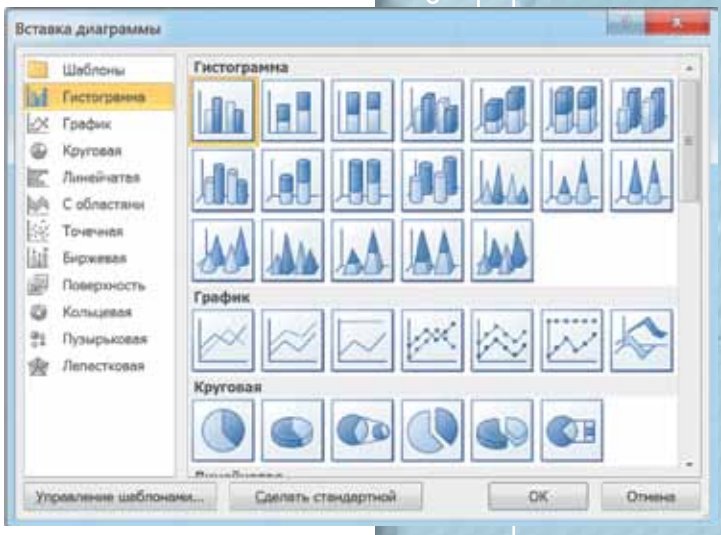


Рис. 32.5

В табличном процессоре *LibreOffice Calc* тип диаграммы выбирают на первом шаге *Мастера диаграмм* (рис. 32.6). Чтобы его вызвать, выбирают инструмент *Диаграмма* на панели инструментов или в меню *Вставка* — команду *Диаграмма*.

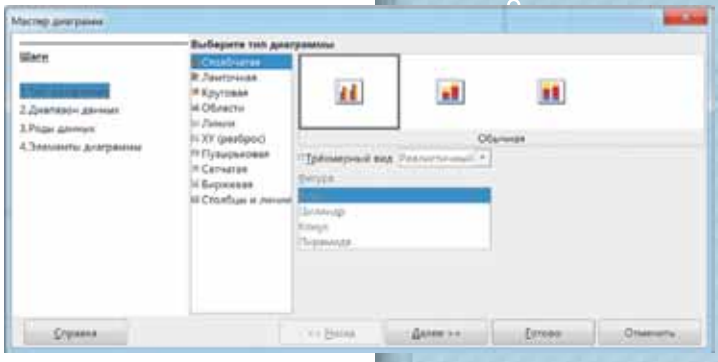


Рис. 32.6

У различных типов диаграмм — свои особенности отображения данных. Для правильного выбора типа диаграмм необходимо понимать назначение каждого из них.

Гистограмма (столбчатая диаграмма) (рис. 32.7) демонстрирует изменение данных за определённый период времени и иллюстрирует соотношение отдельных значений данных. Категории расположены по горизонтали, а значения — по вертикали. Таким образом, уделяется больше внимания изменениям во времени. С помощью гистограммы с накоплением отображается вклад отдельных элементов в общую сумму. Например, по диаграмме с накоплением на рисунке 32.7 можно сделать вывод, что наиболее успешными за указанные 6 лет для сборной Украины были сезоны 2013 и 2014, однако сезон 2015 был успешнее сезонов 2011 и 2012.

Рис. 32.7

Рис. 32.8

Рис. 32.9

Рис. 32.10

Линейчатая диаграмма (**ленточная**, или **горизонтальная гистограмма**) (рис. 32.8) отображает соотношение отдельных компонентов. Категории расположены по горизонтали, а значения — по вертикали. Таким образом, уделяется больше внимания сопоставлению значений и меньше — изменениям во времени. Например, на диаграмме на рисунке 32.8 отображены данные по пяти странам с наибольшим количеством пользователей Интернета в 2014 г. На диаграмме можно увидеть, что хотя население Индии составляет 17,5 % всего населения мира, количество пользователей Интернета в этой стране — 8,33 % от всех пользователей мира, а доля пользователей Интернета США почти вдвое превышает долю этой страны в общей численности населения.

Линейчатая диаграмма с накоплением отображает вклад отдельных элементов в общую сумму.

С помощью **круговой**, или **секторной**, диаграммы (рис. 32.9) иллюстрируется как абсолютная величина каждого элемента ряда данных, так и его доля в общем значении. На круговой диаграмме может представляться только один ряд данных. Такую диаграмму рекомендуется использовать, когда необходимо подчеркнуть некоторый значительный элемент. Например, на рисунке 32.9 представлена диаграмма, иллюстрирующая количество пользователей Интернета по регионам на 1 июля 2013 г. По этой диаграмме можно увидеть, что наибольшую долю — почти половину всех пользователей Интернета — составляют пользователи из Азии.

Для облегчения работы с маленькими долями в основной диаграмме их можно объединить в единый элемент на круговой диаграмме, а затем выделить в отдельную диаграмму рядом с основной (рис. 32.10).

На **графике** — диаграмме типа **линии** — точки, соответствующие данным, соединены линиями (рис. 32.11). На диаграммах такого типа отображаются тенденции изменения данных за равные промежутки времени. Например, на диаграмме на рисунке 32.11 видно, что продажи автомобилей консультантом Ткачуком уменьшались в течение года, особенно во 2 квартале, а другие консультанты увеличили продажи в 4 квартале по сравнению с 3 кварталом.

В **лепестковой**, или **сетчатой**, диаграмме (рис. 32.12) у каждой категории есть собственная ось координат, выходящая из начала координат. Линиями соединены все значения из определённой серии. С помощью лепестковой диаграммы можно сравнить общие значения из нескольких наборов данных. На этой диаграмме ряд данных, ограничивающий наибольшую площадь (сорт 3), характеризует сорт овощей с наибольшим количеством витаминов, а наименьшую площадь (сорт 1) — с наименьшим.

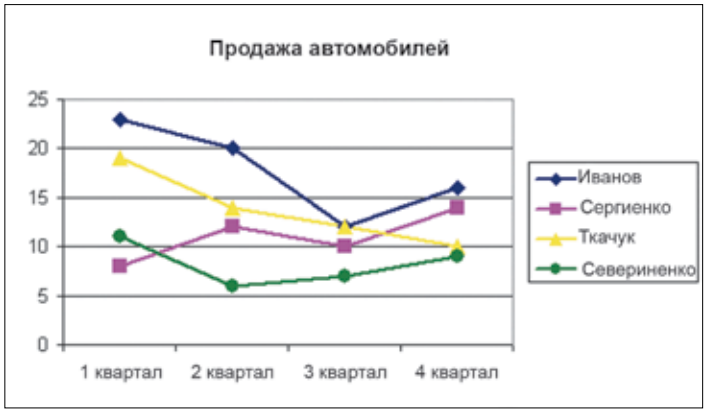


Рис. 32.11

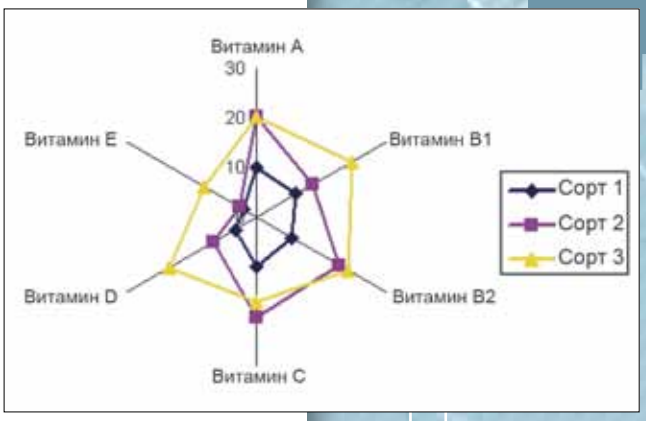


Рис. 32.12

ДЕЙСТВУЕМ

Упражнение 2. Анализ данных, отображающихся на диаграммах различных типов, построенных на основе одних и тех же данных.

Задание. Проанализируйте данные, которые отображены на диаграммах различных типов, построенных по данным таблицы *Олимпийские игры* (рис. 32.13).

	A	B	C	D	E	F
1	Медали Украины на летних Олимпийских играх					
2						
3	Игры	Золото	Серебро	Бронза	Всего	Место
4	1996 Атланта	9	2	12	23	9
5	2000 Сидней	3	10	10	23	21
6	2004 Афины	8	5	9	22	13
7	2008 Пекин	7	5	15	27	11
8	2012 Лондон	6	5	9	20	14
9	Всего	33	27	55	115	

Рис. 32.13

На рисунке 32.14 построена **столбчатая** диаграмма — **вертикальная гистограмма**, на которой показано соотношение золотых, серебряных и бронзовых медалей, завоёванных спортсменами на каждой олимпиаде. Анализируя данные с помощью этой диаграммы, можно сделать вывод, что на олимпиаде 1996 г. в Атланте спортсмены Украины завоевали меньшее количество серебряных медалей, наибольшее количество медалей одного вида — бронзовых — на олимпиаде 2008 г. в Пекине, а на олимпиаде 2000 г. в Сиднее спортсмены завоевали одинаковое количество серебряных и бронзовых медалей.

Рис. 32.14

На рисунке 32.15 изображена нормированная **линейчатая** диаграмма, или **горизонтальная гистограмма**. На нормированной диаграмме общая сумма значений по каждому ряду данных отображается как 100 %. На этой диаграмме можно увидеть, что наибольшее количество всех медалей на всех олимпиадах — это бронзовые, на олимпиадах 1996 г. в Атланте и 2008 г. в Пекине они составляют более половины всех медалей. Также видно, что на олимпиаде 1996 г. в Атланте почти 40 % всех медалей — золотые, а на олимпиадах 2004, 2008 и 2012 гг. спортсмены получили больше золотых медалей, чем серебряных.

Рис. 32.15

Рис. 32.16

Рис. 32.17

Рисунок 32.16 содержит **круговую,** или **секторную,** диаграмму, на которой представлена доля медалей каждого вида в общем количестве. Из анализа данных видно, что за олимпийские игры с 1996 по 2012 г. почти половина всех медалей — бронзовые, но золотых медалей спортсмены завоевали больше, чем серебряных.

Изображённая на рисунке 32.17 **лепестковая,** или **сетчатая**, диаграмма отображает тенденцию количества медалей различных видов на летних олимпиадах с 1996 по 2012 г. Каждой проведённой олимпиаде соответствует ось, на которой обозначено конкретное числовое значение для каждого вида медалей. Видно, что больше всего на всех олимпиадах было завоёвано бронзовых медалей, поскольку у фигуры, образованной на основе данных о бронзовых медалях, наибольшая площадь. В то же время на олимпиаде в Атланте 1996 г. спортсмены завоевали наименьшее количество серебряных медалей, а на олимпиаде в Сиднее 2000 г. ситуация изменилась: количество серебряных медалей сравнялось с количеством бронзовых, а золотых медалей было меньше из всех рассмотренных олимпиад.

3. Как изменять свойства отдельных составляющих диаграммы?

После создания диаграммы её можно размещать в любом месте листа электронной таблицы, удалять, форматировать и редактировать: изменять её размер и внешний вид её отдельных составляющих.

Чаще всего размещают диаграмму непосредственно рядом или после данных, обобщённых на ней. Для удобства восприятия сложные диаграммы нужно делать больше, а простые — меньше по размеру.

Перед перемещением, изменением размера или удалением диаграмму следует выделить. Делается это щелчком в области диаграммы. Перемещение, изменение размеров и удаление диаграммы осуществляется аналогично выполнению этих операций для любого объекта программ пакета *Microsoft Office* или *LibreOffice*. После создания диаграммы её можно форматировать и изменять тип, источник данных, значения параметров диаграммы или место расположения.

В табличном процессоре *Microsoft Excel 2010* это можно сделать с помощью инструментов на вкладках *Конструктор* (рис. 32.18) и *Макет*

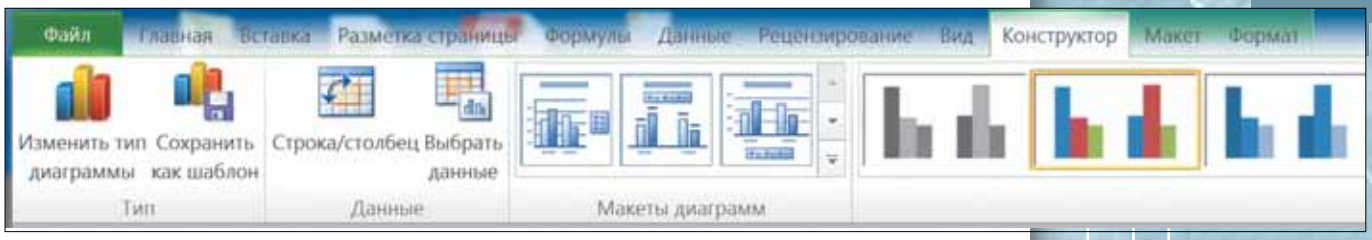


Рис. 32.18

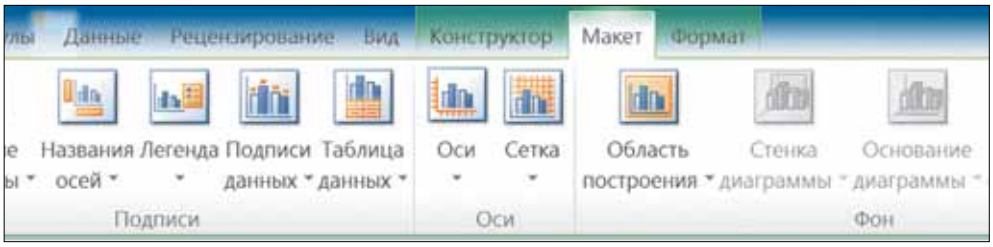


Рис. 32.19

(рис. 32.19), появляющихся в области *Работа с диаграммами* после выделения диаграммы. Можно также щёлкнуть правой кнопкой мыши в свободном от других объектов месте области диаграммы и в контекстном меню выбрать соответствующую команду, которая позволит изменить значения этих свойств. Кроме того, можно изменять значения параметров форматирования любого объекта на диаграмме — например, размер шрифта и другие свойства заголовков и надписей, цвет ряда данных или области построения диаграммы и т. п. Для этого необходимо щёлкнуть правой кнопкой мыши на нужном объекте и выбрать соответствующую команду *Формат рядов данных, Формат легенды, Формат оси, Формат области построения* и т. д.

В табличном процессоре *LibreOffice Calc* после двойного щелчка в области построения диаграммы панель инструментов меняет свой вид (рис. 32.20). Инструменты *Тип диаграммы*  , *Форматировать избранное*  и другие используются для изменения значений свойств диаграммы.

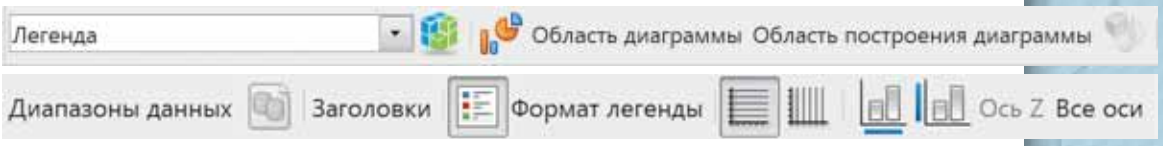


Рис. 32.20

ДЕЙСТВУЕМ

Упражнение 3. Изменение типа диаграммы, источника данных для её построения и значений параметров форматирования некоторых её объектов.

Задание. Измените источник данных диаграммы *Рост населения*, содержащейся в папке *Электронные таблицы*, для построения диаграммы так, чтобы на ней были отображены данные только для двух частей света — Европы и Азии. Измените тип диаграммы на линейчатую, цвет заливки столбцов диаграммы и её области по образцу (рис. 32.21).

1. В папке *Электронные таблицы* откройте файл *Рост населения*.
2. Выделите диаграмму, представленную на листе, для этого щёлкните на ней мышью. Перейдите на вкладку *Конструктор* на ленте (рис. 32.18)

Рис. 32.21

и выберите инструмент *Выбрать данные* в *Microsoft Excel 2010* (на панели инструментов выберите *Диапазоны данных* в *LibreOffice Calc*). Выделите в таблице новый диапазон ячеек *A3:E5*, чтобы на диаграмме отображались данные только по двум частям света — Европе и Азии. Нажмите кнопку *OK*.

3. На вкладке *Конструктор* выберите инструмент *Тип диаграммы*. Выберите тип *Линейчатая* (*Ленточная*). Нажмите кнопку *OK*.
4. Измените цвета некоторых рядов данных по образцу (рис. 32.21), для этого последовательно щёлкните правой кнопкой мыши на соответствующем ряду данных, выберите команду *Формат рядов данных*. В открывшемся окне выберите *Заливка* в левой части окна, установите переключатель *Сплошная заливка*, выберите нужный цвет и нажмите кнопку *Закрыть*.
5. Аналогично измените цвет фона области построения диаграммы.
6. Добавьте подписи данных на диаграмме, для этого на вкладке *Макет* (рис. 32.19) выберите инструмент *Подписи данных* и значение *У вершины, снаружи*.
7. Сохраните результаты в файле с тем же именем в папке *Табличный процессор* своей структуры папок.

4. Как настроить значения параметров страницы перед печатью электронной таблицы?

Как и для текстовых документов, при создании электронной таблицы устанавливаются значения параметров страницы по умолчанию: ориентация страницы — книжная или альбомная, размеры левого, правого, верхнего и нижнего полей, размер листа бумаги и т. п. Для электронных таблиц, созданных на основе шаблонов, такие значения параметров могут отличаться от применяемых к новой пустой книге. Как и в текстовом процессоре, в табличном процессоре *Microsoft Excel 2010* с помощью команды *Файл/Создать* (в текстовом процессоре *LibreOffice Calc* с помощью команды *Файл/Создать/Шаблоны*) можно выбрать шаблон для создания новой книги (рис. 32.22), содержащий уже некоторые данные, оформление и значения параметров страницы.

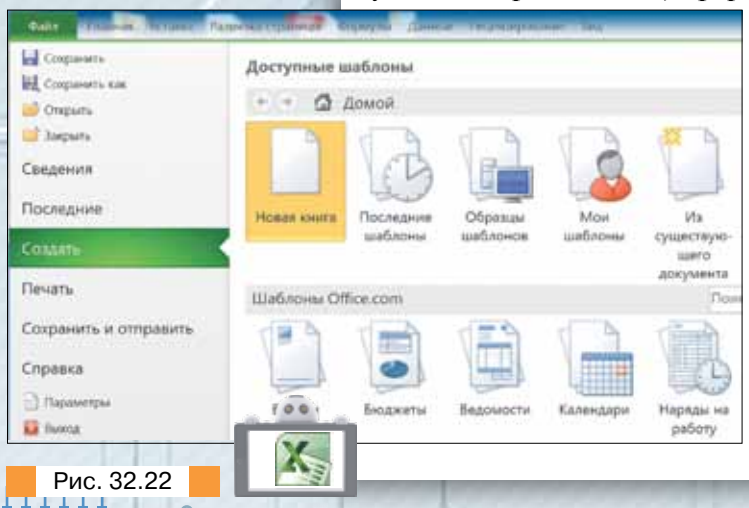


Рис. 32.22

Например, новая книга, созданная на основе шаблона *Журнал кровяного давления*, имеет следующие значения параметров страницы: ориентация страницы — книжная; левое, верхнее, правое и нижнее поля — по 1,3 см.

Чтобы изменить значения параметров страницы в табличном процессоре *Microsoft Excel 2010*, можно воспользоваться инструментами на вкладке *Разметка страницы* (рис. 32.23). Все значения параметров страницы можно просмотреть или изменить также в окне *Параметры страницы* (рис. 32.24), вызываемом с помощью кнопки в правой нижней части группы *Параметры страницы*.

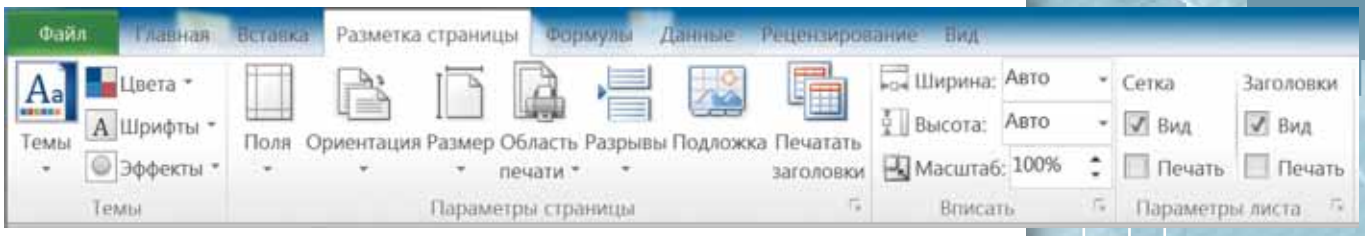


Рис. 32.23

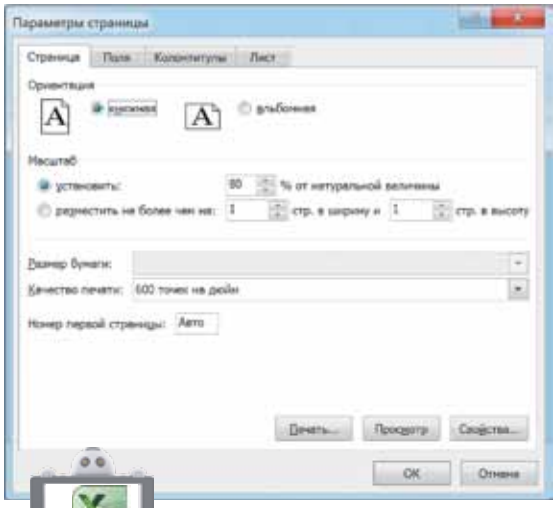


Рис. 32.24

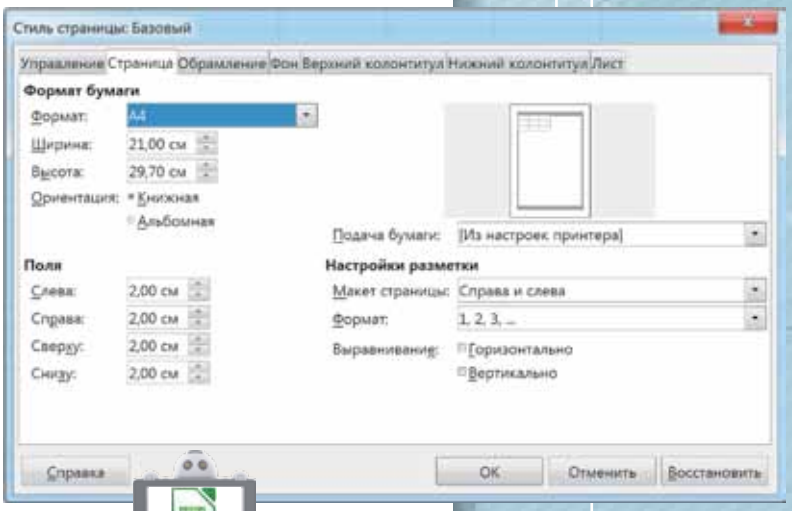


Рис. 32.25

Аналогичное окно *Стиль страницы: Базовый* в табличном процессоре *LibreOffice Calc* (рис. 32.25) вызывают с помощью команды *Формат/ Страница*.

Иногда вся таблица может не поместиться на страницу формата А4, на которой её предполагается печатать. Например, может не поместиться один столбец по ширине и две строки по высоте. Поэтому при подготовке к печати электронной таблицы нужно учитывать масштаб отображения таблицы. Изменяя масштаб, можно разместить всю таблицу на одном листе.

5. Как напечатать электронную таблицу?

После ввода данных в таблицу, оформления, выполнения вычислений, создания диаграмм и других объектов, настройки параметров страницы можно напечатать электронную таблицу.

В табличном процессоре *Microsoft Excel 2010*, как и в текстовом процессоре *Microsoft Word 2010*, для печати используют команду *Файл/Печать* и указывают значения параметров печати: выбирают принтер, количество копий для печати, порядок печати нескольких листов и при необходимости изменяют значения параметров страницы (рис. 32.26).

Значения параметров можно изменить, если открыть соответствующие списки (рис. 32.26). Кроме того, можно выбрать один из режимов пользовательского масштабирования, при котором автоматически будет установлен масштаб отображения для размещения таблицы на одной странице либо всех столбцов или

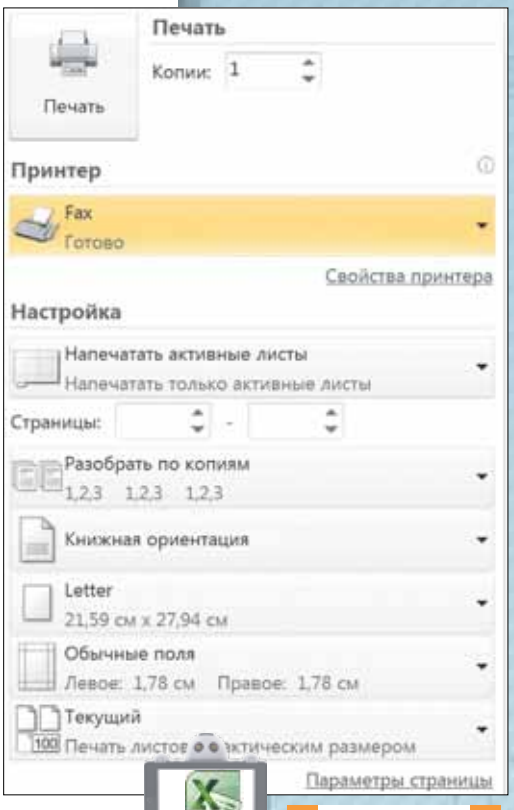


Рис. 32.26

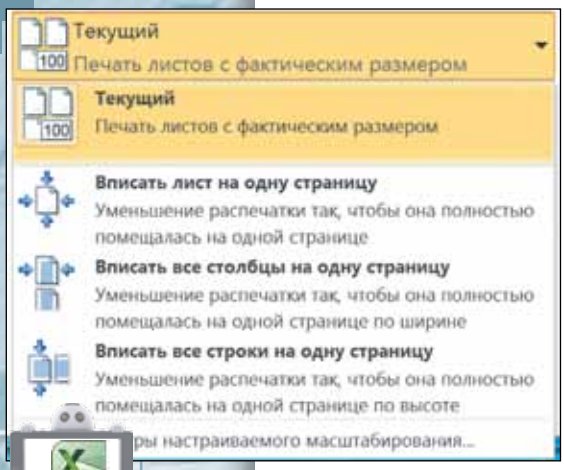


Рис. 32.27

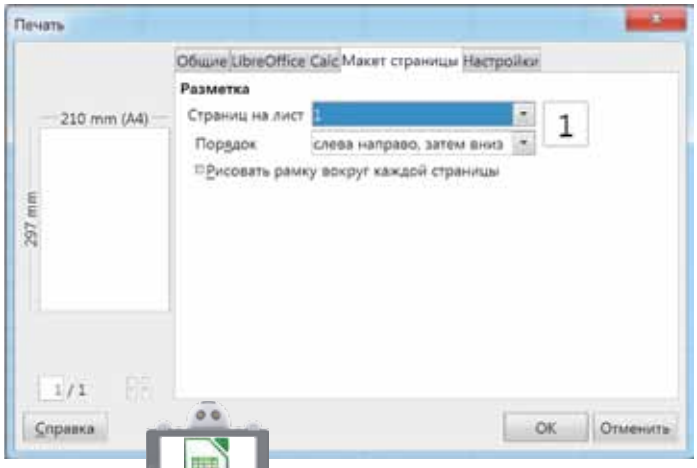
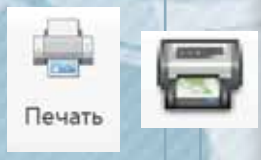


Рис. 32.28

строк на одной странице (рис. 32.27). После установки значений всех необходимых параметров следует нажать кнопку *Печать*.

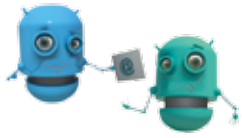
В табличном процессоре *LibreOffice Calc* параметры печати и страницы собраны на разных вкладках окна *Печать* (рис. 32.28), открываемого с помощью команды *Файл/Печать* или кнопки *Печать* на панели инструментов. Для печати таблицы нажимают кнопку *OK*.



ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 32* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ



1. Проанализируйте, диаграммы каких типов используются в ваших учебниках по географии, истории, биологии. Постройте соответствующую таблицу частотности использования диаграмм определённого типа. Проанализируйте и обсудите в паре, для отображения каких данных используются диаграммы в учебниках.
2. Спросите у родителей и знакомых, какой тип диаграмм они используют чаще всего. Обсудите причины и сделайте вывод.
3. По ключевому слову *график* найдите в Интернете различные изображения. Обсудите аргументы правильного выбора соответствующих типов диаграмм. Одним предложением опишите назначение найденной диаграммы и укажите тип диаграммы, которым можно заменить найденный.
4. Обсудите, какие трудности могут возникнуть при чтении диаграммы, если диаграмма не содержит:
   - названия диаграммы;
   - легенду;
   - подписей данных.
5. Задайте друг другу вопросы по данным диаграммы (рис. 32.29). Оцените ответ каждого на заданные вопросы.

6. Сравните, что общего и какие отличия есть у табличного и текстового процессора для:
- настройки параметров страницы;
- печати документа.

По результатам сравнения постройте диаграмму Венна.

7. Для оформления школьной газеты вы решили построить диаграммы. Обсудите, какой тип диаграммы нужно использовать для данных, отображающих:
1) количество обращений учеников школы в школьную библиотеку;
2) показатели изменения высоты растения в течение недели в опыте о влиянии света на рост растений;
3) процент занятости учащихся в работе школьных кружков;
4) данные о среднем использовании Интернета учениками разных классов;
5) тенденцию изменения отношения учеников разных классов к экологическим проблемам до просмотра выступления школьной группы «За чистую окружающую среду» и после него;
6) вклад каждого класса в общую сумму средств, собранную во время благотворительной ярмарки.

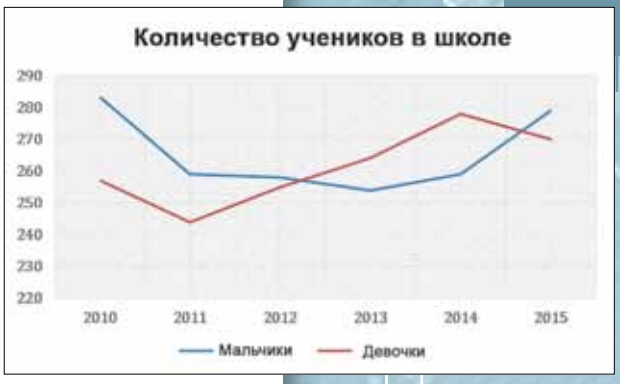


Рис. 32.29

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. Рассмотрите диаграмму, представленную на рисунке 32.30. Проанализируйте диаграмму по следующей схеме:
а) Почему выделенные комиксы считаются успешными?
б) На производство какого комикса по бюджету было израсходовано наибольшую сумму денег; наименьшую сумму?
в) Соответствует ли действительности гипотеза (аргументируйте свое мнение, опираясь на диаграмму):
- новейшие комиксы получили самый большой бюджет;
- бюджет, израсходованный на производство комиксов, увеличивается в каждом году, начиная с 1989 г.;
- с каждым годом сборы от экранизации комиксов увеличиваются;
- бюджет на изготовление второй версии самых успешных комиксов всегда больше (меньше);
- сборы от экранизации вторых версий самых успешных комиксов всегда больше (меньше).

г) Во сколько раз денежные сборы от экранизации комиксов превышают израсходованный на них бюджет?
д) От экранизации какого комикса получены самые большие денежные сборы?

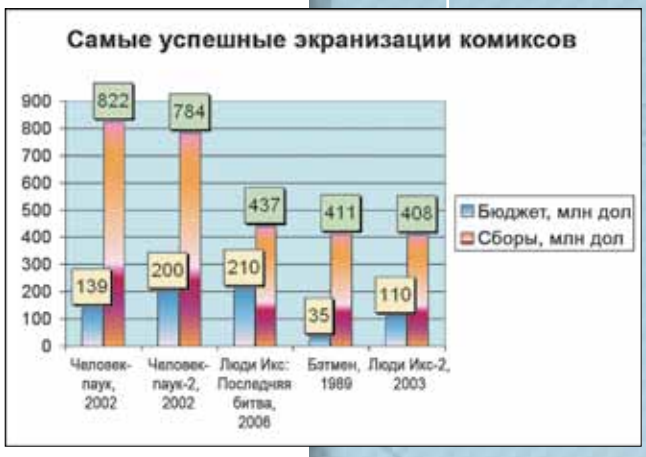


Рис. 32.30

2. В таблице 32.1 представлены сведения об уровне занятости (% от общего количества) населения соответствующей возрастной группы по месту жительства. Постройте диаграмму,

Таблица 32.1

Возрастная группа	15–24	25–29	30–39	40–49	50–59	60–70
Городское население	34,8	77,7	82,4	80,4	59,2	13,7
Сельское население	43,1	72,7	79,3	78,6	65,5	37,3

по которой можно отследить тенденцию изменения доли занятого населения соответствующей возрастной группы по месту жительства.

3. Известен химический состав воздуха: азот — 78,08 %, кислород — 20,94 %, инертные газы — 0,94 %, диоксид углерода — 0,04 %. Отобразите графически данные о химическом составе воздуха, выберите целесообразный тип и вид диаграммы.

4. По данным диаграммы (рис. 32.30) создайте в табличном процессоре таблицу с заголовком *Успешные экранизации комиксов*, которую сохраните в файле *Комиксы* в папке *Табличный процессор* своей структуры папок. По данным созданной таблицы постройте диаграмму, как показано на рисунке 32.30, разместите её на отдельном листе книги *Комиксы* и сохраните результаты работы.

5. На основе опроса о преимуществах и недостатках инструментов в онлайновых обучающих играх получены следующие результаты:

Преимущества:

Переход на уровни | Очки/баллы | Онлайн-результаты | Отслеживание прогресса | Возможность коммуникации

Недостатки:

Соревнование с друзьями | Виртуальные подарки | Быть героем истории | Аватарки | Виртуальная валюта

Представьте все данные на одной презентации. Выберите тип диаграммы и способ выделения преимуществ и недостатков рассматриваемых инструментов.

6. Создайте алгоритм для печати трёх копий электронной таблицы, представьте его графически.

ИССЛЕДУЕМ

1. Определите, какие действия необходимо выполнить, чтобы построить комбинированную диаграмму по образцу (рис. 32.10) для данных, содержащихся в файле *Города Украины* в папке *Электронные таблицы*.

2. Определите, какие свойства для ряда данных пузырьковой диаграммы в табличном процессоре *Microsoft Excel 2010* можно менять. Для этого откройте в папке *Электронные таблицы* файл *Пузырьковая диаграмма*, выделите только числовые данные для построения диаграммы (без заголовков строк и столбцов), постройте пузырьковую объёмную диаграмму. Установите формат ряда данных: значениям соответствует площадь пузырьков, масштаб пузырьков — 300. При необходимости выберите на вкладке *Конструктор* инструмент *Строка/столбец* для получения результата, как на рисунке 32.31.

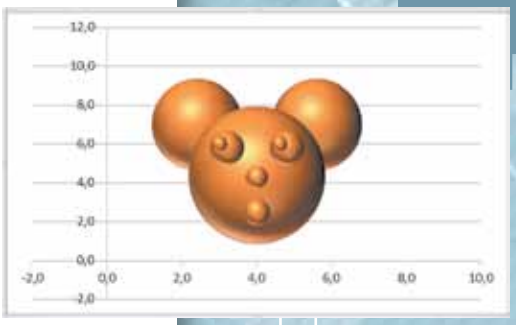


Рис. 32.31

33. СОРТИРОВКА И ФИЛЬТРАЦИЯ ДАННЫХ В ТАБЛИЦАХ. ПРОМЕЖУТОЧНЫЕ ИТОГИ

ВСПОМНИТЕ:

- как вводить данные в табличном процессоре;
- данные каких типов могут содержать ячейки электронной таблицы;
- как вычислить сумму, максимальное, минимальное и среднее значения для диапазона ячеек электронной таблицы.

ВЫ УЗНАЕТЕ:

- как сортировать данные в электронной таблице;
- как и для чего в электронных таблицах используются фильтры;
- по каким правилам создаются условия для расширенного фильтра;
- что такое промежуточные итоги и как ими пользоваться.

ИЗУЧАЕМ

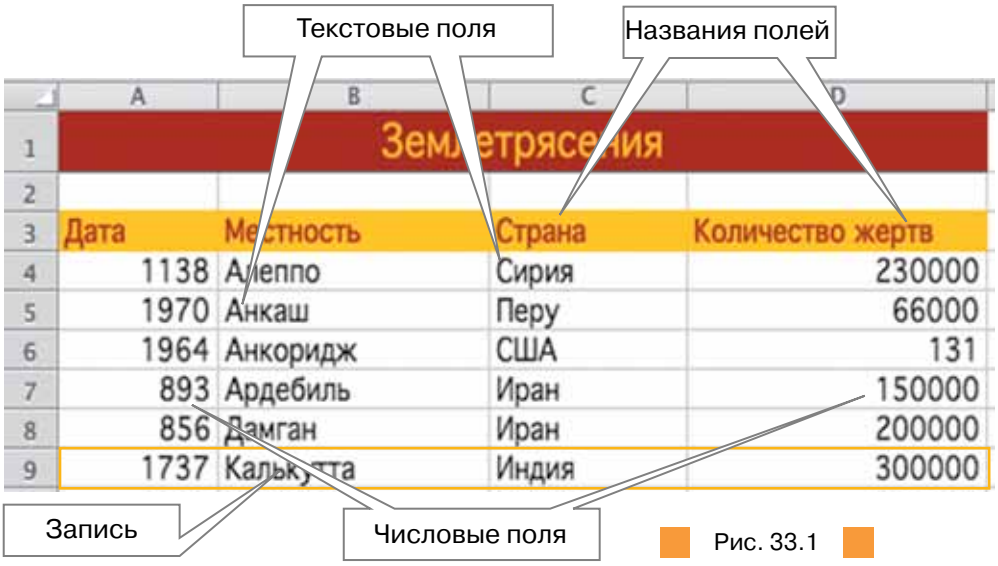
1. Как сортировать данные в электронной таблице?

Для анализа и поиска нужных данных в таблицах, где данных очень много, используются встроенные средства, одно из них — **упорядочение**, или **сортировка**, данных.

Сортировка любых данных выполняется для ускорения поиска. Например, если данные в столбце, содержащем фамилии, упорядочены по алфавиту, то в нём искать данные о человеке с конкретной фамилией легче, чем в неупорядоченном списке. Но когда список данных достаточно большой, например, в нём много родственников или однофамильцев, найти нужные данные сложнее.

Данные связанного диапазона в табличном процессоре можно сортировать по значению содержимого одного или нескольких столбцов по возрастанию или по убыванию. В процессе сортировки не только будут менять своё место в таблице ячейки того столбца, по которому происходит сортировка, но и будут переставляться записи, содержащие данные о каждом объекте таблицы.

Ячейки с данными в столбце называются **полем**, а строки — **записями**. Каждая запись содержит данные об одном объекте таблицы, например, о землетрясении: когда оно произошло, в каком городе и стране это случилось и к какому количеству жертв привело (рис. 33.1). Верхняя строка таблицы при этом содержит названия полей.



	A	B	C	D
1	Землетрясения			
2				
3	Дата	Местность	Страна	Количество жертв
4	1138	Алеппо	Сирия	230000
5	1970	Анкаш	Перу	66000
6	1964	Анкоридж	США	131
7	893	Ардебиль	Иран	150000
8	856	Дамган	Иран	200000
9	1737	Калькутта	Индия	300000

Рис. 33.1

Сортировка происходит в два этапа: выделение поля таблицы для сортировки и непосредственно сортировка. При работе с данными всей электронной таблицы или с отдельными её полями не обязательно выделять всю таблицу или весь столбец. Достаточно выделить любую ячейку поля, по значениям которого в первую очередь будут упорядочиваться данные всей таблицы.

Далее в табличном процессоре *Microsoft Excel 2010* следует воспользоваться одним из способов доступа к инструментам сортировки:

1) на вкладке *Главная* в группе *Редактирование* открыть список инструмента *Сортировка и фильтр* (рис. 33.2);

2) соответствующие инструменты размещены также на вкладке *Данные* в группе *Сортировка и фильтр* (рис. 33.3);

3) в контекстном меню, появляющемся при щелчке на любой ячейке поля, по которому необходимо упорядочить данные, выбрать команду *Сортировка* (рис. 33.4).

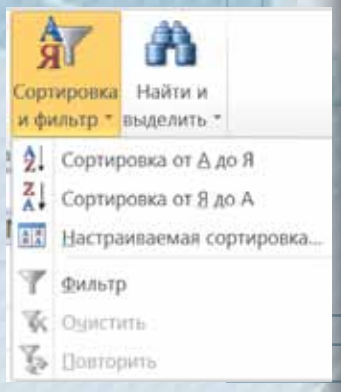


Рис. 33.2

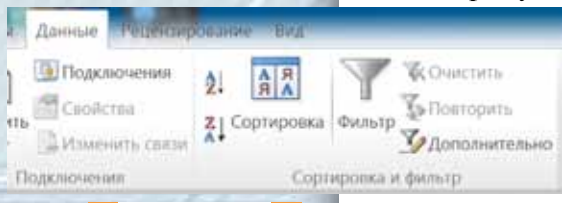


Рис. 33.3

Затем следует воспользоваться одним из инструментов сортировки по одному полю по возрастанию или убыванию, или выбрать *Настраиваемую сортировку*, если следует упорядочить данные таблицы по нескольким полям одновременно.

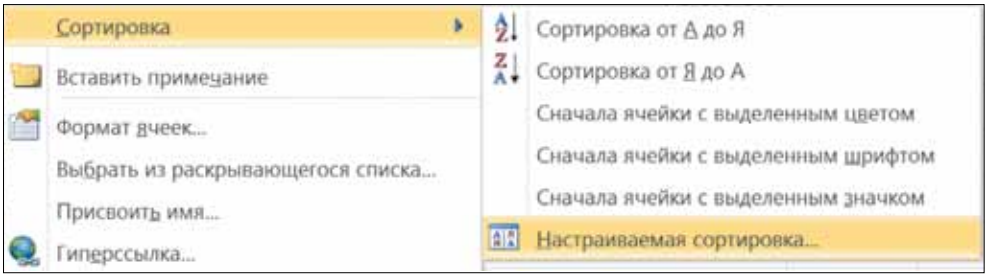


Рис. 33.4

Инструменты  и  изменяют своё название в зависимости от типа значений в поле, по которому происходит сортировка:

- текстовые значения — *Сортировка от А до Я* и *Сортировка от Я до А*, при этом сортировка осуществляется по алфавиту или в обратном порядке;
- числовые значения — *Сортировка от минимального к максимальному* и *Сортировка от максимального к минимальному*;
- значение типа дата и время — *Сортировка от старых к новым* и *Сортировка от новых к старым*.

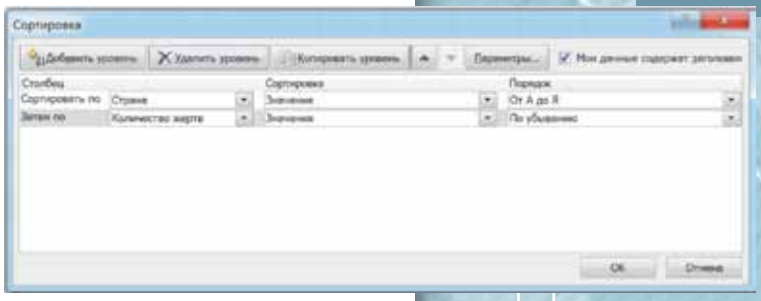


Рис. 33.5

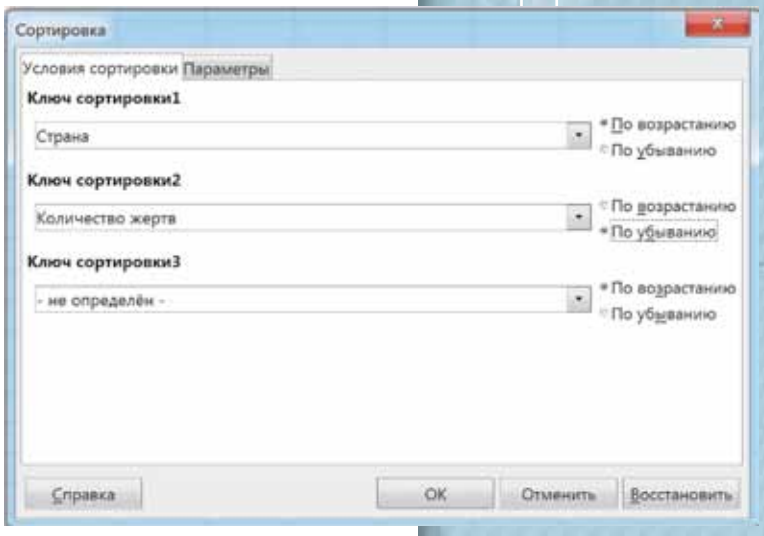


Рис. 33.6

Если выбрать команду *Настраиваемая сортировка*, открывается окно *Сортировка* (рис. 33.5), в котором пользователь может последовательно добавить несколько полей для сортировки с помощью кнопки *Добавить уровень*. Выбрать поля для сортировки можно с помощью списков в окне *Сортировка*. Дополнительно для каждого поля, по которому будет происходить сортировка, следует отметить порядок сортировки — по возрастанию или по убыванию. Сортировку по нескольким полям используют, если одно из полей содержит группы одинаковых значений, тогда в пределах каждой такой группы данные будут сортироваться по второму полю. Например, если в таблице о землетрясениях выполнить сортировку по двум полям — *Страна* и *Количество жертв*, то для каждой группы записей, для которых название страны одинаково, данные будут отсортированы по количеству жертв.

В табличном процессоре *LibreOffice Calc* сортировка по одному полю осуществляется с помощью инструментов *Сортировка по возрастанию*  или *Сортировка по убыванию*  на панели инструментов *Стандартная*. Для сортировки по нескольким полям одновременно используется команда меню *Данные/Сортировка* или инструмент *Сортировка*  на панели инструментов. При этом открывается окно *Сортировка*, в котором можно выбрать названия до трёх полей и указать порядок сортировки по каждому из них (рис. 33.6).

ДЕЙСТВУЕМ

Упражнение 1. Сортировка данных таблицы.

Задание. В таблице *Землетрясения*, находящейся в папке *Электронные таблицы*, выполните сортировку данных по дате по возрастанию; затем по количеству жертв по убыванию; по двум полям: по странам в алфавитном порядке и по количеству жертв по убыванию. Ответьте на заданные вопросы.

1. В папке *Электронные таблицы* загрузите файл *Землетрясения*. Назовите имена полей таблицы (рис. 33.1).

	A	B	C	D
1	Землетрясения			
2				
3	Дата	Местность	Страна	Количество жертв
4	1556	Шаньси	Китай	800000
5	1976	Тянь-Шань	Китай	655000
6	1737	Калькутта	Индия	300000
7	1138	Алеппо	Сирия	230000
8	856	Дамган	Иран	200000

Рис. 33.7

2. Выделите любую ячейку поля *Дата* и выберите инструмент *Сортировка по возрастанию*. Ответьте на вопрос: *Какие мощные землетрясения зафиксированы до н. э., какие в XVIII в., а какие — в XX в.?*

3. Выделите любую ячейку поля *Число жертв* и выберите инструмент *Сортировка по убыванию* (рис. 33.7). Ответьте на вопрос: *В каких странах мощные землетрясения вызвали наибольшее количество жертв?*

4. Выделите любую ячейку поля *Страна* и выберите инструмент *Настраиваемая сортировка (Сортировка)*. В окне *Сортировка* в первом списке уже установлено имя нужного поля, нажмите кнопку *Добавить уровень* и во втором списке выберите поле *Число жертв*. Выберите для него порядок сортировки *От максимального к минимальному* (*по убыванию*) (рис. 33.5, 33.6). Подтвердите выполнение операции, нажав кнопку *ОК*. Проанализируйте результаты и ответьте на вопросы:
1. В каких странах происходило несколько землетрясений?
2. Какое землетрясение привело к наибольшему количеству жертв в Китае?
3. Какое землетрясение привело к наибольшему количеству жертв в Японии?

5. Сохраните файл с тем же именем в папке *Табличный процессор* своей структуры папок.

2. Как и для чего в электронных таблицах используются фильтры?

Кроме сортировки данных, для быстрого поиска в таблице данных, соответствующих некоторым условиям, используют фильтры.

Фильтры являются средством быстрого выделения из таблицы набора данных, соответствующих заданным условиям.

После применения фильтра на экране остаются только записи, соответствующие заданным условиям, другие строки скрываются. **Фильтрация** — процесс применения к электронной таблице правил отбора данных для отображения на экране. Условия фильтрации могут быть простыми и составными. После анализа результатов фильтрации можно отобразить на экран все данные исходной таблицы.

При работе с табличным процессором для фильтрации данных можно воспользоваться средствами *Автофильтр* и *Расширенный фильтр*.

Автофильтр используют, если необходимо выбрать из таблицы данные с определённым значением ячеек или сформировать условия, поддерживаемые этим средством. Автофильтр можно применить последовательно к нескольким полям, однако условия, создаваемые по разным полям, будут связываться только логической операцией И. Составное условие по одному полю может быть создано с использованием логических операций И либо ИЛИ, но количество объединяемых условий ограничено.

Для использования всех инструментов для работы с фильтрами в *Microsoft Excel 2010* выбирают вкладку *Данные* в группе *Сортировка и фильтр* (рис. 33.3).

Чтобы воспользоваться *Автофильтром*, необходимо выделить любую ячейку таблицы, которую нужно фильтровать, и выбрать инструмент *Фильтр*.

Интересно

Фильтрация в табличном процессоре напоминает процесс фильтрации для очистки воды, газов от твёрдых частиц. Например, перед тем как вода из водоёма попадёт в водопровод, она проходит несколько этапов очистки, на каждом из которых осуществляется отбор определённой группы примесей. Фильтрация позволяет организовать «сито», через которое можно «просеять» частицы или данные, соответствующие определённым условиям.

Раскрывающиеся списки

	A	B	C	D	E	F	G	H
1	Фамилия	Имя	Дата рождения	Возраст	Рос	Пол	Цвет глаз	Увлечение
2	Демиденко	Евгений	12.09.2006	10	145	мальчик	серые	теннис
3	Денисенко	Вадим	15.10.2006	10	144	мальчик	серые	борьба
4	Шаповалова	Мария	23.12.2006	10	132	девочка	карие	танцы

Рис. 33.8

После включения режима *Автофильтр* (это встроенное средство отбора) на экране во всех полях таблицы отображаются кнопки раскрывающихся списков (рис. 33.8).

При нажатии кнопки со стрелкой в каждом поле раскрывается список возможных значений, а также команды для сортировки данных в этом поле и формирования условий фильтрации (рис. 33.9). В зависимости от типа данных в каждом поле команды для создания условий фильтрации могут отличаться. С помощью текстовых фильтров можно создавать условия для поиска текстовых значений, начинающихся или заканчивающихся на определённый символ или набор символов, содержащих или не содержащих некоторый текстовый фрагмент (рис. 33.9). Числовые фильтры используются для создания условий с операторами сравнения — больше, меньше и т. д., а также для нахождения значений, больших или меньших среднего, отображения указанного количества наибольших или наименьших значений и т. п. С помощью фильтров дат можно найти данные, дата которых — перед или после заданной даты, содержится в заданном диапазоне и т. п.

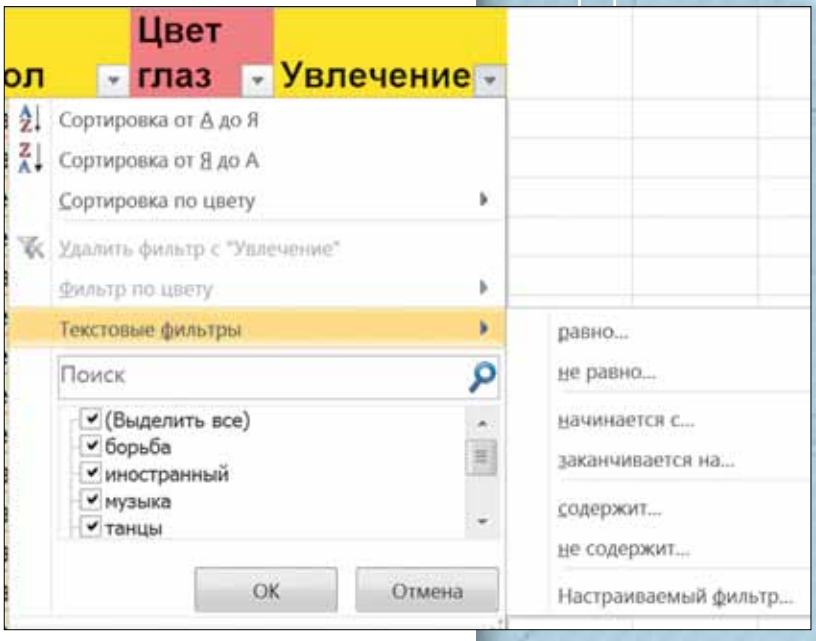


Рис. 33.9

Для выбора данных, совпадающих с определённым значением, следует в списке уникальных значений отметить флажками значения, которые необходимо искать (рис. 33.9). По умолчанию выбран режим *Выделить всё* — его можно выключить и выбирать только нужные значения.

Если нужно найти записи, удовлетворяющие составным условиям, состоящим из двух простых условий для значений одного поля текстового или числового типа, то необходимо выбрать из списка фильтров *Настраиваемый фильтр*. На экране отображается окно *Пользовательский автофильтр*, в котором можно:

- указать одно или два простых условия;
- для каждого простого условия выбрать операцию (равно, больше, меньше и т. д.) и значение для сравнения;
- указать логическую операцию, используемую для составного условия из указанных простых, — И, ИЛИ (рис. 33.10).

Рис. 33.10

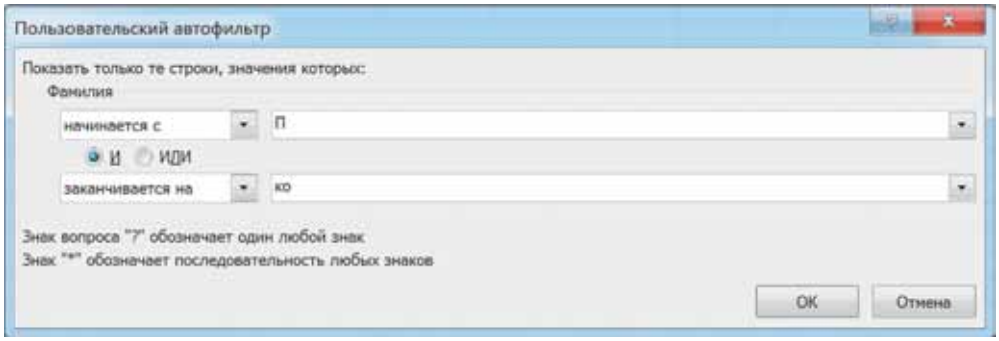
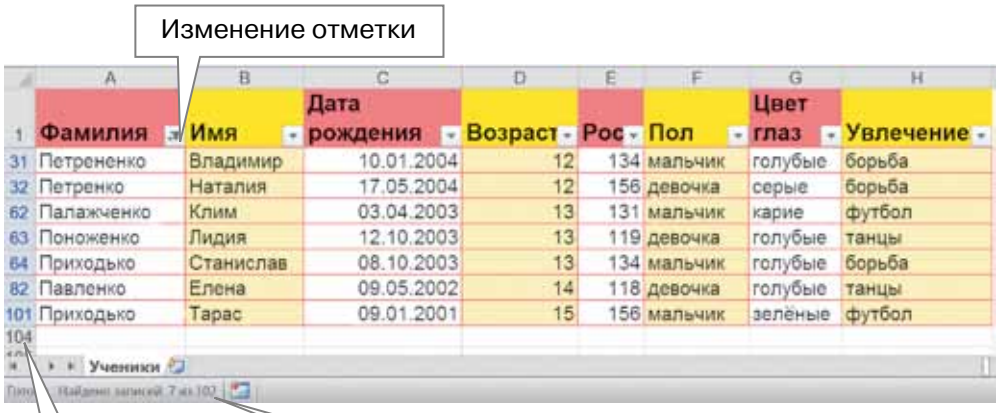


Рис. 33.11



Результаты отбора сразу же отображаются на экране, что определяется по трём признакам (рис. 33.11):

- номерам строк отфильтрованных записей — некоторые номера пропускаются, а номера остальных отображаются синим цветом;
- содержимому строки состояния — отображается сообщение о найденном количестве записей после выполнения фильтрации;
- изменению отметки  на кнопке поля, по значениям которого осуществлялся отбор данных.

Для отображения всех данных списка на экран необходимо выбрать инструмент *Очистить* в группе *Сортировка и фильтрация* на вкладке *Данные*.

Таким образом, алгоритм использования средства *Автофильтр* можно сформулировать так:

1. Выделить любую ячейку таблицы, содержащую данные.
2. Вызвать средство *Автофильтр* одним из способов.
3. Сформировать условия поиска данных с помощью встроенных средств. Для создания составных условий нужно воспользоваться командой *Настраиваемый фильтр*.
4. Проанализировать отображённые данные.
5. После анализа данных отменить действие фильтра — использовать инструмент *Очистить*.

В *LibreOffice Calc* для работы с фильтрами используют команду *Автофильтр* в меню *Данные* (рис. 33.12). Также *Автофильтр* можно вызвать с

помощью одноимённого инструмента  на панели инструментов *Стандартная*.

В отличие от *Microsoft Excel 2010,* в *LibreOffice Calc* средства создания условий при использовании автофильтра одинаковы для данных всех типов, с их помощью можно выбирать значения, совпадающие с заданными значениями в выбранном поле, упорядочивать данные, находить записи, содержащие пустые или непустые значения, отображать на экране указанное количество наибольших или наименьших значений с помощью команды *10 первых* (рис. 33.13). Для создания более сложных условий можно выбрать в этом списке или в меню *Данные/Ещё фильтры* команду *Стандартный фильтр*. Такие условия можно формировать в окне *Стандартный фильтр* (рис. 33.14), которое напоминает окно *Пользовательский автофильтр* в *Microsoft Excel 2010*, однако позволяет объединять до четырёх простых условий.

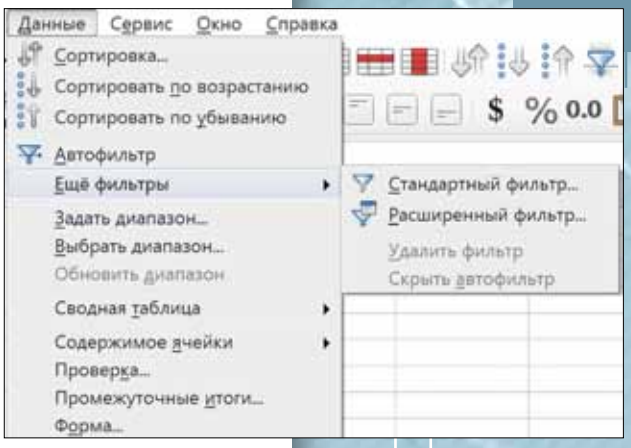


Рис. 33.12

ДЕЙСТВУЕМ

Упражнение 2. Использование условия при применении *Автофильтра*.

Задание. С помощью средства *Автофильтр* найдите в электронной таблице, содержащейся в файле *Ученики*, количество и список мальчиков, фамилии которых начинаются с *П* и заканчиваются на *-ко* и которые занимаются футболом.

1. В папке *Электронные таблицы* откройте файл *Ученики*.
2. Выделите любую ячейку таблицы, содержащую данные.
3. На вкладке *Данные* в группе *Сортировка и фильтр* выберите инструмент *Фильтр*.
4. В списке в поле *Фамилия* выберите команду *Текстовые фильтры/Настраиваемый фильтр*.
5. В окне *Пользовательский автофильтр* из перечня операций выберите *Начинается с* (рис. 33.10, 33.14), а рядом в поле ввода введите символ *П*. Выберите логический оператор И для создания составного условия и запишите второе условие — *Заканчивается на ко*. Нажмите кнопку *ОК*. На экране отображается отфильтрованный список.
6. В списке в поле *Пол* из списка значений выберите *мальчик*, а затем в списке в поле *Интересы* выберите *футбол*.
7. Убедитесь, что на экране отображены записи, соответствующие условию задания. Определите их количество.
8. Для отображения сведений обо всех учениках на экране выберите инструмент *Очистить* на вкладке *Данные* (выполните команду *Данные/Ещё фильтры/Удалить*).

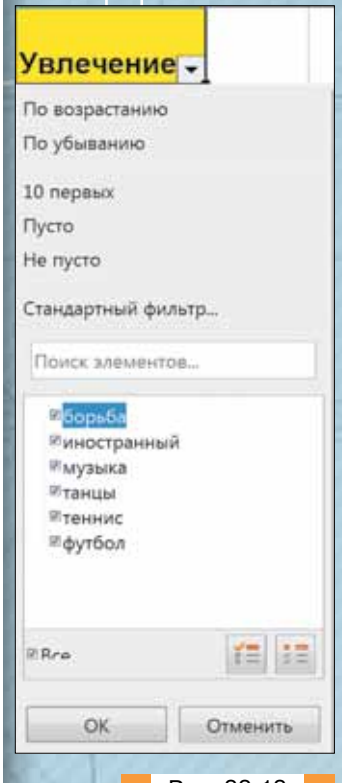


Рис. 33.13

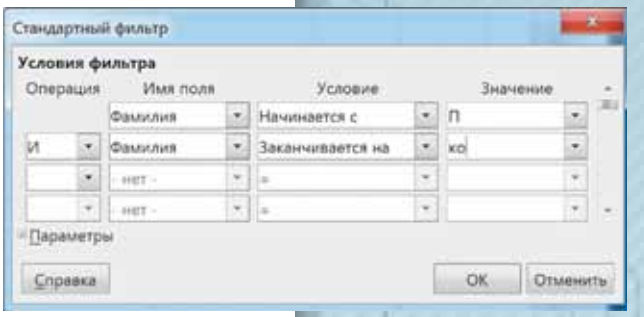


Рис. 33.14

3. По каким правилам создаются условия для расширенного фильтра?

Расширенный фильтр используется при необходимости сформировать составные условия поиска, не поддерживаемые средством *Автофильтр*. Это может быть в случае, если условия по разным полям следует объединить логической операцией ИЛИ, либо если для

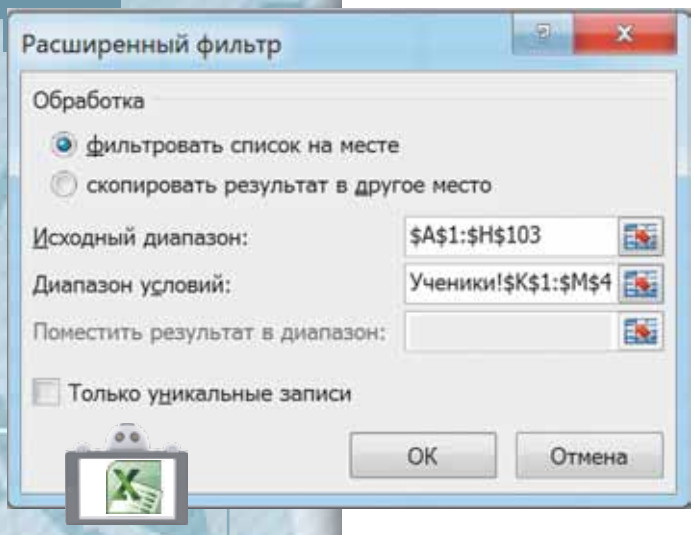


Рис. 33.15, *а*

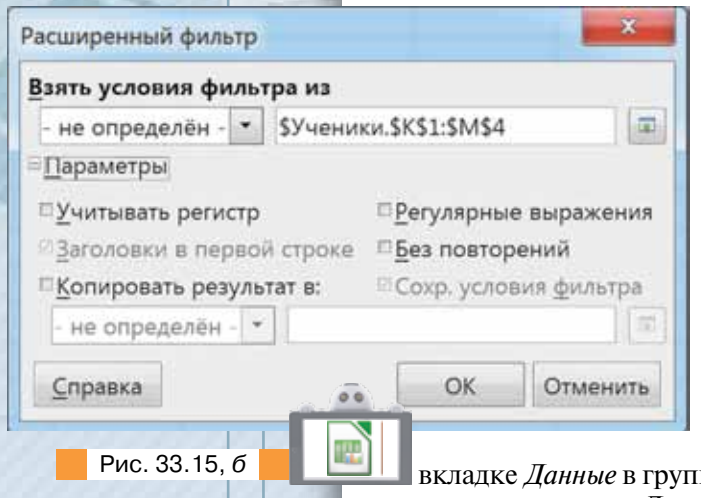


Рис. 33.15, *б*

одного поля нужно описать составное условие, содержащее больше простых условий, чем поддерживается *Пользовательским автофильтром* (два для *Microsoft Excel 2010* и четыре для *LibreOffice Calc*).

В отличие от автофильтра, который сначала вызывают, а затем формируют условия фильтрации с помощью встроенных средств, прежде чем вызвать расширенный фильтр, пользователь должен в некотором диапазоне таблицы сформировать условия фильтрации по следующим правилам:

1. Для того чтобы отфильтровать данные таблицы с помощью расширенного фильтра, необходимо, чтобы у полей таблицы были заголовки, т. е. столбцы имели названия.
2. Между диапазоном, в котором будут записываться условия, и таблицей данных должна быть хотя бы одна пустая строка или один пустой столбец.
3. Скопируйте из таблицы заголовки столбцов, в которых осуществляется поиск, и вставьте их в первую пустую строку диапазона условий отбора. Диапазон условий не обязательно должен содержать названия всех полей таблицы с данными.
4. Введите в строки под заголовками диапазона условий необходимые условия отбора.
5. Составные условия, которые должны быть связаны логической операцией И, следует записать в одной строке. Условия, записанные в разных строках, связываются логической операцией ИЛИ.
6. Выделите любую ячейку таблицы с данными.
7. Вызовите *Расширенный фильтр* — в *Microsoft Excel 2010* выберите инструмент *Дополнительно* на вкладке *Данные* в группе *Сортировка и фильтр* (в *LibreOffice Calc* выполните команду меню *Данные/Ещё фильтры/Расширенный фильтр*).
8. Для того чтобы отобразить результаты фильтрации, скрыв при этом ненужные строки, следует в окне *Расширенный фильтр* в *Microsoft Excel 2010* установить переключатель в положение *Фильтровать список на месте* (рис. 33.15, *а*), в *LibreOffice Calc* — не включать флажок *Копировать результат в* (рис. 33.15, *б*).
9. Введите в поле *Диапазон условий* (*Взять условия фильтра из*) ссылку на диапазон с записанными условиями отбора, включая заголовки столбцов.

Дополнительно

ДЕЙСТВУЕМ

Упражнение 3. Использование расширенного фильтра.

Задание. С помощью расширенного фильтра в файле *Ученики* найдите записи, содержащие сведения о мальчиках, увлекающихся музыкой, девочках, занимающихся иностранным языком, а также всех учащихся в возрасте 10 лет — независимо от их интересов.

1. В папке *Электронные таблицы* откройте файл *Ученики*.
2. Начиная с ячейки *K1*, создайте диапазон условий в соответствии с заданием (рис. 33.16).

K	L	M
Увлечение	Пол	Возраст
		10
музыка	мальчик	
иностранный	девочка	

Рис. 33.16

	A	B	C	D	E	F	G	H
1	Фамилия	Имя	Дата рождения	Возраст	Рост	Пол	Цвет глаз	Увлечение
2	Балеев	Сергей	14.05.2006	10	143	мальчик	серые	танцы
4	Бочаров	Игорь	15.03.2001	15	124	мальчик	серые	музыка
8	Верешага	Надежда	29.12.2002	14	147	девочка	карие	иностранный
12	Галушко	Сергей	16.08.2003	13	145	мальчик	зелёные	музыка
15	Гладченко	Фёдор	20.01.2001	15	123	мальчик	серые	музыка
20	Гурченко	Марина	28.02.2002	14	148	девочка	серые	иностранный
25	Демиденко	Евгений	12.09.2006	10	145	мальчик	серые	теннис
26	Денисенко	Вадим	15.10.2006	10	144	мальчик	серые	борьба
33	Иваненко	Наталия	17.12.2003	13	148	девочка	голубые	иностранный
41	Красная	Елена	09.08.2002	14	149	девочка	голубые	иностранный
43	Курис	Елена	10.02.2003	13	147	девочка	голубые	иностранный
45	Любченко	Ольга	12.08.2005	11	137	девочка	зелёные	иностранный
63	Петрова	Светлана	01.03.2002	14	145	девочка	голубые	иностранный
64	Петрова	Наталия	23.10.2002	14	146	девочка	карие	иностранный
87	Супрун	Анна	12.12.2005	11	136	девочка	серые	иностранный
96	Федорченко	Наталия	25.02.2002	14	137	девочка	зелёные	иностранный
99	Шаповалова	Мария	23.12.2006	10	132	девочка	карие	танцы
104								

Ученики

Рис. 33.17

3. Выделите ячейку *A1*. Выберите инструмент *Дополнительно* на вкладке *Данные* в группе *Сортировка и фильтр* (выполните команду *Данные/Ещё фильтры/Расширенный фильтр*).
4. Убедитесь, что в диалоговом окне *Расширенный фильтр* (рис. 33.15, *а*) в области *Исходный диапазон* указано *$A$1:$H$103* (иначе выделите этот диапазон ячеек).
5. В диалоговом окне *Расширенный фильтр* установите текстовый курсор в область *Диапазон условий* и выделите в таблице диапазон, содержащий созданные условия: *K1:M4*. Адрес этого диапазона с абсолютными ссылками и названием листа будет отображён в области *Диапазон условий*. Нажмите кнопку *OK*.
6. Определите, сколько записей таблицы соответствует указанным условиям (рис. 33.17).
7. Сохраните результаты в файле с именем *Расширенный фильтр* в папке *Табличный процессор* своей структуры папок.

4. Что такое промежуточные итоги и как ими пользоваться?

Вы уже умеете применять стандартные функции для вычисления суммы, максимального, минимального и среднего значений некоторых диапазонов ячеек. Если таблица содержит данные, для которых по некоторым полям можно создать группы с одинаковыми значениями, в табличном процессоре можно воспользоваться встроенным средством для быстрого вычисления итоговых значений в полях, содержащих числовые данные. Чтобы создать такие группы, необходимо упорядочить данные таблицы по тем полям, которые содержат одинаковые значения. После этого можно воспользоваться средством *Промежуточный итог* для добавления итоговых значений для указанных числовых полей.

Например, в таблице, содержащей данные о мощных землетрясениях, с помощью такого средства можно быстро вычислить общее количество жертв в каждой стране отдельно. Если данные в поле *Страна* упорядочены, то мы имеем чёткие группы, после каждой из которых можно вставить итоговую запись.

Рис. 33.18

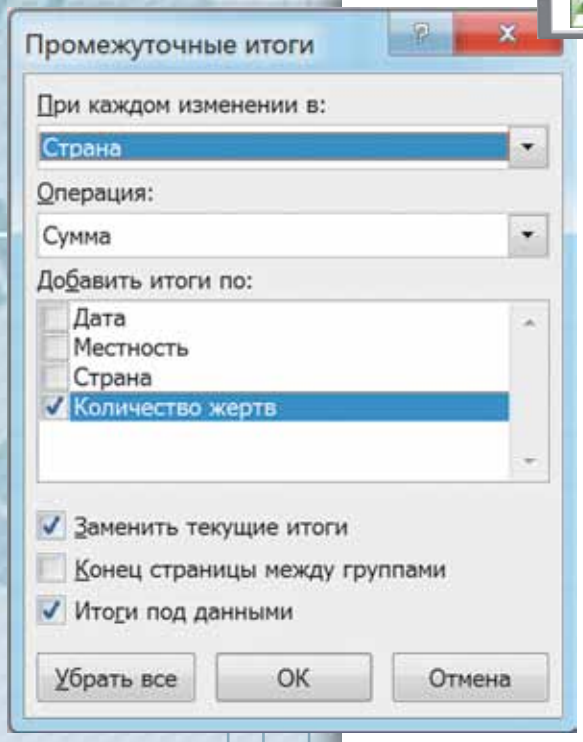


Рис. 33.19

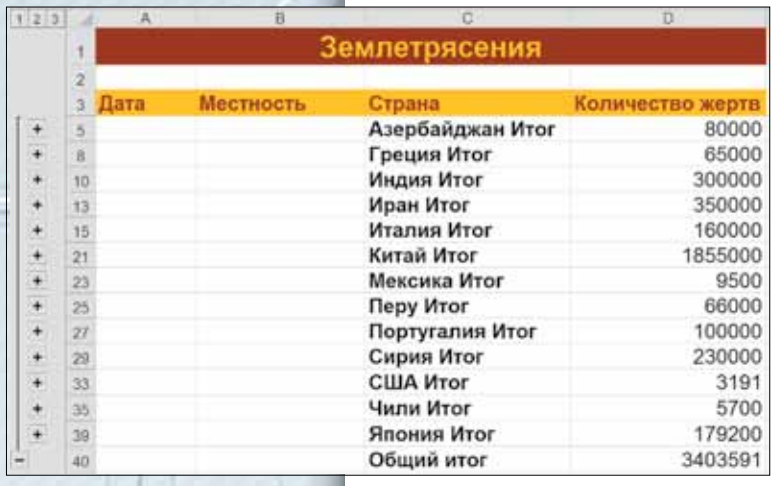
Итак, первое условие для автоматической вставки итогов — это сортировка данных в поле, по которому создаются группы. После этого следует использовать средство добавления промежуточных итогов.

В *Microsoft Excel 2010* для этого необходимо:

- На вкладке *Данные* в группе *Стуктура* выбрать инструмент *Промежуточный итог* (рис. 33.18).
- В окне *Промежуточные итоги* (рис. 33.19) из списка *При каждом изменении в* выбрать поле, в котором выполнена сортировка и образованы группы записей с одинаковыми значениями.
- В списке *Операция* выбрать функцию, которую необходимо использовать при вычислении промежуточных итогов: сумма, количество, среднее значение, максимум, минимум и др.
- В поле *Добавить итоги по* указать поля, по которым должны вычисляться промежуточные итоги. Выбирать следует только поля, содержащие числовые данные.
- При необходимости снять флажок *Итоги под данными* для отображения строки с итогами над соответствующими данными.
- Если отдельные группы записей должны быть расположены и выведены на печать на разных страницах, — установить флажок параметра *Конец страницы между группами*.
- Закрыть окно, нажав кнопку *OK*.

Аналогично можно добавить вложенные промежуточные итоги. Для этого нужно сначала упорядочить данные в таблице по нескольким полям одновременно и последовательно использовать инструмент *Промежуточные итоги,* создавая поочерёдно итоги каждого из уровней. После вставки итогов первого уровня на следующем шаге в окне *Промежуточный итог* следует снять флажок *Заменить текущие итоги*.

После добавления промежуточных итогов слева от таблицы появляется структура (рис. 33.20), с помощью которой можно скрыть или отобразить строки с данными для отдельных промежуточных итогов.



	A	B	C	D
1	Землетрясения			
2				
3	Дата	Местность	Страна	Количество жертв
5			Азербайджан Итог	80000
8			Греция Итог	65000
10			Индия Итог	300000
13			Иран Итог	350000
15			Италия Итог	160000
21			Китай Итог	1855000
23			Мексика Итог	9500
25			Перу Итог	66000
27			Португалия Итог	100000
29			Сирия Итог	230000
33			США Итог	3191
35			Чили Итог	5700
39			Япония Итог	179200
40			Общий итог	3403591

Рис. 33.20

Для этого можно воспользоваться значками – и + для различных уровней каждой из групп данных. Скрыть или отобразить промежуточные итоги определённого уровня можно с помощью кнопок с номерами уровней 1 2 3 в верхней части такой структуры:

- 1 — вывод только общих итогов;
- 2 — вывод общих и промежуточных итогов;
- 3 и ниже — вывод полного списка.

Для удаления итогов, а вместе с ними и структуры, нужно открыть диалоговое окно *Промежуточные итоги* и нажать кнопку *Убрать все*. Чтобы заменить текущие итоги

новыми, получаемыми по другой формуле или для других полей, следует задать в этом окне нужные параметры и установить флажок *Заменить текущие итоги*.

Промежуточные итоги будут автоматически удалены при пересортировке списка. При этом на экран будет выведено предупреждение.

В табличном процессоре *LibreOffice Calc* добавление промежуточных итогов происходит по такому же алгоритму. После сортировки данных таблицы следует выполнить команду *Данные/Промежуточные итоги* и задать все необходимые свойства в окне *Промежуточные итоги* (рис. 33.21). Для таблицы, упорядоченной по нескольким полям, можно создать вложенные итоги с помощью вкладок, определяющих уровни: *1-я группа, 2-я группа, 3-я группа*.

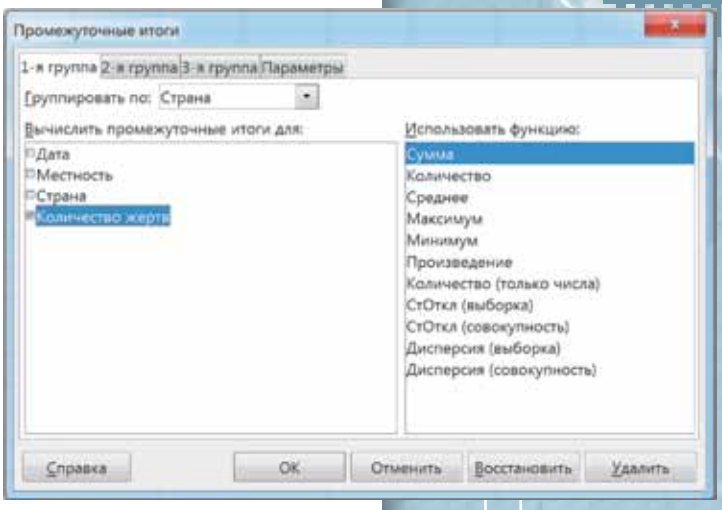


Рис. 33.21

ДЕЙСТВУЕМ

Упражнение 4. Создание промежуточных итогов.

Задание. В файле *Землетрясения* добавьте промежуточные итоги, отображающие общее количество жертв мощных землетрясений в каждой стране.

1. В папке *Электронные таблицы* откройте файл *Землетрясения*.
2. Выделите ячейку *C3*, содержащую заголовок поля *Страна*. На вкладке *Данные* в группе *Сортировка и фильтр* выберите инструмент *Сортировка от А до Я* (на панели инструментов *Стандартная* выберите инструмент *Сортировка по возрастанию*).
3. На вкладке *Данные* в группе *Структура* выберите инструмент *Промежуточный итог* (выполните команду *Данные/Промежуточные итоги*). В диалоговом окне *Промежуточные итоги* (рис. 33.19, 33.21) в списке *При каждом изменении в:* выберите поле *Страна*, в списке *Операция*: выберите *Сумма*, в списке *Добавить итоги по* включите флажок для поля *Количество жертв* (для других полей флажки должны быть выключены). Нажмите кнопку *ОК*.
   В таблицу после каждой группы стран будут добавлены итоговые записи. В левой части окна появится иерархическая структура, которая позволит отображать все записи вместе с итогами или только итоговые записи (рис. 33.22).
4. Нажмите на каждой кнопке – 2-го уровня в области структуры, что позволит скрыть исходные записи таблицы, а отображать только итоговые записи (рис. 33.20).
5. Сохраните результаты в файле с именем *Итоги* в папке *Табличный процессор* своей структуры папок.

	A	B	C	D
1	Землетрясения			
2				
3	Дата	Местность	Страна	Количество жертв
4	1667	Шемаха	Азербайджан	80000
5			**Азербайджан Итог**	80000
6	-856	Коринф	Греция	45000
7	-464	Спарта	Греция	20000
8			**Греция Итог**	65000
9	1737	Калькутта	Индия	300000
10			**Индия Итог**	300000
11	893	Ардебиль	Иран	150000
12	856	Дамган	Иран	200000
13			**Иран Итог**	350000
14	1908	Мессина	Италия	160000
15			**Италия Итог**	160000
16	1920	Каньшу	Китай	200000
17	1731	Пекин	Китай	100000
18	1976	Тянь-Шань	Китай	655000
19	1038	Чихли	Китай	100000
20	1556	Шаньси	Китай	800000
21			**Китай Итог**	1855000

Рис. 33.22

ОБСУЖДАЕМ

Обсудите вопросы, содержащиеся в файле *Тема 33* в папке *Обсуждаем*. Ознакомиться с этими вопросами можно также с помощью QR-кода.

РАБОТАЕМ В ПАРАХ

1. Для чего в жизни применяют сортировку данных? По очереди называйте примеры — выиграет тот, кто назовёт пример последним.
2. Зачем использовать фильтры в быту? Приведите как можно больше примеров. Назовите три аргумента, подтверждающих, что действие бытовых фильтров подобно действию фильтрации данных с помощью средств табличного процессора.
3. В каких случаях при создании промежуточных итогов целесообразно, чтобы флажок *Заменить текущие итоги* был включён, а когда его целесообразно выключить? Приведите примеры. Обсудите в парах.
4. Что общего между автофильтром и автозаполнением ячеек электронных таблиц? Ответ аргументируйте. Обсудите в парах.
5. В папке *Электронные таблицы* откройте файл *Страны*.
   а) Сформулируйте три вопроса на сортировку и отбор данных из таблицы, удовлетворяющих определённым критериям.
   К примеру:
   - В каких странах название столиц начинается с буквы *С*?
   - Площадь каких стран превышает площадь Украины?
   - Население каких стран меньше населения Украины?

   б) Выполните отбор данных для ответа на заданные вопросы.
   в) Задайте сформулированные вопросы другой паре. Ответьте на вопросы, предложенные другой парой, выполнив на основе заданной таблицы соответствующую фильтрацию данных.
   г) Проверьте ответы друг друга. Обсудите полученные результаты.
   д) Отсортируйте данные стран по плотности населения на квадратный километр. Для этого вставьте перед полем *Столица* столбец *Плотность населения* и проведите необходимые вычисления.

РАБОТАЕМ САМОСТОЯТЕЛЬНО

1. В папке *Электронные таблицы* откройте файл *Поезда*.
   а) Отсортируйте данные последовательно по направлению, наличию билетов, стоимости, времени отправления.
   б) С помощью средства *Автофильтр* найдите сведения о поездах:
   - на которые есть билеты и которые отправляются ежедневно;
   - которые отправляются в Запорожье и на которые есть билеты;
   - которые отправляются во Львов позже всего;
   - которые отправляются в Варшаву по самому дешёвому тарифу.

   в) Создайте диапазон условий и примените расширенный фильтр для поиска сведений о поездах, отправляющихся ежедневно или время отправления которых — позже 17:30.
2. В папке *Электронные таблицы* откройте файл *Турист*.
   а) Отсортируйте данные таблицы одновременно по странам, в них — по городам, в них — по стоимости номеров за сутки.

б) С помощью средства *Автофильтр* найдите в таблице данные о:
- всех отелях Парижа с категорией три звезды (1 *** или 2 ***) и предлагающих завтрак и ужин;
- всех гостиницах для поездки в Вашингтон (США), где включено питание;
- 10 самых дешёвых отелях в Париже;
- всех турах, включающих авиабилеты и шведский стол для питания;
- всех отелях Рима, стоимость проживания за сутки в которых находится в пределах 200–300 € и где предусмотрены завтрак и ужин.

в) С помощью промежуточных итогов найдите среднюю стоимость номера за сутки в каждой стране и в каждом городе.

3. В папке *Электронные таблицы* откройте файл *Ученики*.

а) С помощью средства *Автофильтр* найдите данные о:
- всех девочках, увлекающихся танцами, в возрасте от 12 до 14 лет. Результаты фильтрации скопируйте на новый лист;
- всех мальчиках, занимающихся футболом, рост которых больше 130 см. Результаты фильтрации скопируйте на новый лист;
- всех учениках с карими глазами в возрасте 14 лет, фамилия которых начинается с буквы *В*. Результаты фильтрации скопируйте на новый лист.

б) Выполните сортировку по возрастанию по полю *Возраст*. Добавьте промежуточные итоги, отображающие средний рост учеников каждого возраста.

ИССЛЕДУЕМ

Определите, каким будет результат добавления промежуточных итогов в случае, если данные электронной таблицы не упорядочены ни по одному из полей. Сделайте вывод.

ПОЛЕЗНЫЕ ССЫЛКИ

Сортировка данных в диапазоне или таблице:
https://support.office.com/ru-ru/article/Сортировка-данных-в-диапазоне-или-таблице-62d0b95d-2a90-4610-a6ae-2e545c4a4654
Вставка промежуточных итогов:
https://support.office.com/ru-ru/article/Вставка-промежуточных-итогов-в-список-данных-листа-7881d256-b4fa-4f81-b71e-b0a3d4a52b3a
Фильтрация данных с помощью автофильтра:
https://support.office.com/ru-ru/article/Краткое-руководство-фильтрация-данных-с-помощью-автофильтра-08647e19-11d1-42f6-b376-27b932e186e0

34. ПРАКТИЧЕСКАЯ РАБОТА 16

СОРТИРОВКА ДАННЫХ В ТАБЛИЦАХ. АВТОМАТИЧЕСКИЕ И РАСШИРЕННЫЕ ФИЛЬТРЫ

ВСПОМНИТЕ

- Как сортировать данные электронной таблицы по одному и нескольким полям;
- как с помощью средства *Автофильтр* задавать условия поиска данных в электронной таблице;
- по каким правилам создавать диапазон условий для расширенного фильтра.

СОЗДАЙТЕ

В своей структуре папок создайте папку *Практическая работа 16.*

ПОМНИТЕ

Выполняя практические задания, соблюдайте правила безопасности жизнедеятельности при работе с компьютером!

Задание 1. Самые высокие водопады мира (*12 баллов*)

В электронной таблице *Водопады*, находящейся в папке *Электронные таблицы* и содержащей сведения о самых высоких водопадах мира, примените фильтры и сортировку данных для быстрого поиска ответов на вопросы, указанные в тетради или сформулированные учителем.

Задание 2. Самые длинные реки Украины (*12 баллов*)

В электронной таблице *Реки Украины*, находящейся в папке *Электронные таблицы* и содержащей сведения о самых длинных реках Украины, примените фильтры и сортировку данных для быстрого поиска ответов на вопросы, указанные в тетради или сформулированные учителем.

Задание 3. Олимпиада по информатике (*14 баллов*)

В электронной таблице *Олимпиада 2016,* находящейся в папке *Электронные таблицы* и содержащей сведения о результатах городского этапа олимпиады по информатике в Киеве, примените фильтры и сортировку данных для быстрого поиска ответов на вопросы, указанные в тетради или сформулированные учителем. Добавьте промежуточные итоги для вычисления средней суммы баллов, которые набрали ученики каждого района.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Фамилия, имя, отчество	Район	Школа	Класс	Аудитория	ПК	1 numbers	2 triangle	3 graph	4 camps	5 roads	6 umbrella	Сумма баллов	Место	Отбор
2	Гречка Артём Витальевич	Соломенский	52	5	512А	14						108	108	3	
3	Вахитов Антон Владимирович	Дарницкий	НЗ	7	213	1	70	80		80			230	1	
4	Заводник Владислав Александрович	Голосеевский	179	7	319	13		100		100			200	1	
5	Абдулаев Андрей Анатольевич	Дарницкий	НЗ	7	123	1	45	30		80		32	187	2	
6	Скоробогатько Игорь Александрович	Дарницкий	НЗ	7	224	1	90	40		20		12	162	2	
7	Рудик Евгения Александровна	Печерский	171	7	213	11							0		
8	Затилюк Андрей Александрович	УФМЛ		8	321	2	60	100	20	40		24	244	1	
9	Руденко Анастасия Вячеславовна	Печерский	171	8	319	7	90	60		10		59	219	1	
10	Тычковский Алексей Русланович	УФМЛ		8	123	3	60					118	178	2	
11	Ткаченко Роман Андреевич	Дарницкий	НЗ	8	505	1	90	80					170	2	
12	Черныш Дмитрий Валентинович	Соломенский	52	8	512В	6	60	40	10	40		12	162	2	

35. РЕШЕНИЕ КОМПЕТЕНТНОСТНЫХ ЗАДАЧ

1. Задание «Переезд в Киев»

Летом вы с семьёй собираетесь переехать в Киев, поскольку там работают ваши родители. Вы уже выбрали вуз для дальнейшего обучения, остаётся выбрать школу для младшего брата, который учится в 8 классе, имеет способности и опыт составления компьютерных программ. Вы вместе с братом решили найти в Киеве школу, ученики которой показали лучшие результаты на олимпиаде по информатике за последний год. Аргументом для родителей в правильном выборе школы может стать диаграмма количества призёров в пяти учебных заведениях с наиболее высокими результатами. Постройте такую числовую диаграмму на основе полученной электронной таблицы.

1. Выберите форму представления найденных данных, которая будет удобна для просмотра родителями: презентация, электронное письмо, текстовый документ, проект, подготовленный на языке программирования. Учтите, что родителям следует сообщить URL-адреса сайтов двух лучших, по вашему мнению, учебных заведений и аргументы в их пользу.
2. Заполните таблицу выполнения задания — файл *Таблица-олимпиада* в папке *Компетентностные задачи*.
3. На электронный адрес учителя пришлите решение задачи: архив с разработанной электронной таблицей; архив представления результатов для родителей, таблицу выполнения задания.

2. Задание «Детский праздник»

У вашего двоюродного брата скоро день рождения — ему исполнится 7 лет. Его родители обратились к вам за помощью в создании сметы для проведения детского праздника.

Брат хотел бы пригласить 11 друзей. Угощать гостей планируется дома двумя видами пирожных, соком и мороженым, которые можно купить в одном из супермаркетов. Комнату можно украсить воздушными шариками самим или заказать оформление в агентстве «Весёлый клоун», о котором родители наслышанны и хотели бы получить информацию о нём для приглашения клоуна на праздник.

1. Составьте смету праздника с учётом экономии для семейного бюджета. Для этого создайте электронную таблицу и выполните вычисления с помощью формул или разработайте проект в одной из сред программирования. Сравните доли каждого вида затрат (продукты, развлечения, оформление) в общей сумме расходов на праздник, построив соответствующий тип диаграммы для обоснования и принятия решения родителями брата.
2. Заполните таблицу выполнения задания — файл *Таблица-праздник* в папке *Компетентностные задачи*.
3. На электронный адрес учителя пришлите решение задачи: архив или проект электронной таблицы, содержащей вычисления на основе формул, и построенную диаграмму; таблицу выполнения задания.

3. Задание«Фермер»

Фермер, изучив спрос на ягодные культуры в своём регионе, планирует 70 % своего земельного участка площадью 2 га отвести под выращивание клубники. Для этого он собирается обратиться к руководству банка с письмом о предоставлении ему соответствующего кредита.

1. Рассчитайте количество кустов определённого сорта клубники и нужную сумму на закупку рассады при условии, что за первую сотню кустов покупатель платит полную стоимость, за каждую следующую сотню стоимость уменьшается на 1 % от предыдущей стоимости.

Рассаду фермер планирует закупить в интернет-магазине «Дом и сад» из расчёта 40–60 тыс. кустов на 1 га. Выберите программу для выполнения задания: табличный процессор или среду программирования.

2. Напишите письмо для получения фермером кредита от банка, в котором представьте нужные данные, выводы и аргументируйте их. Добавьте в текст письма изображение выбранного сорта клубники.
3. Заполните таблицу выполнения задания — файл *Таблица-фермер* в папке *Компетентностные задачи*.
4. На электронный адрес учителя пришлите решение задачи: текстовый документ; архив электронной таблицы или проекта, разработанного в среде программирования; таблицу выполнения задания.

4. Задание «Поездка на автомобиле»

Семья из Днепра планирует в течение года путешествовать по разным городам Украины на автомобиле. Найдите расстояния от Днепра до Киева, Львова, Харькова, Одессы, Херсона, Винницы и ориентировочную стоимость бензина А-95. Определите необходимые технические характеристики автомобиля — объём бака для топлива и расход топлива на 100 км, если семья путешествует на автомобиле Nissan Note с двигателем 1,6 л и автоматической коробкой передач.

1. Создайте электронную таблицу, в которой укажите расстояния от Днепра до указанных городов, рассчитайте количество необходимого топлива и стоимость топлива до каждого города, а также по формуле с использованием логической функции определите, для поездки в какие города необходимо дополнительно заправлять автомобиль по пути. Постройте диаграмму, на которой отобразите расстояния и стоимость поездки до каждого из городов.
2. Выберите город, в который, по вашему мнению, следует поехать в первую очередь, и аргументируйте ваш выбор.
3. Заполните таблицу выполнения задания — файл *Таблица-поездка* в папке *Компетентностные задачи*.
4. На электронный адрес учителя пришлите решение задачи: файл электронной таблицы; архив проекта электронного путеводителя; таблицу выполнения задания.

5. Задание «Стартап»

Вы вместе с друзьями решили летом начать свой проект «Летний кинотеатр» для сбора средств на восстановление старого замка. Для этого на стене замка вы планируете демонстрировать молодёжные фильмы, учебное и научно-популярное видео и т. п. Разработайте необходимые информационные материалы и расчёты расходов и доходов от проекта.

1. С целью привлечения инвестиций и поддержки со стороны бизнеса и местной власти разработайте текстовый документ — брошюру о своём проекте. В брошюре на отдельных страницах представьте идею проекта, его задачи, основные шаги реализации, таблицу запланированных сеансов и соответствующую смету. Добавьте к брошюре иллюстрации замка.
2. Спланируйте и разработайте электронную таблицу или проект в среде программирования, с помощью которых можно фиксировать расходы для подготовки к сеансу (прокат аппаратуры, печать объявлений, средства для уборки мусора после сеанса), благотворительные пожертвования на ремонт замка и рассчитать полученную прибыль.
3. Заполните таблицу выполнения задания — файл *Таблица-стартап* в папке *Компетентностные задачи*.
4. На электронный адрес учителя пришлите решение задачи: текстовый документ; архив проекта для расчёта или электронной таблицы; таблицу выполнения задания.

36. ВЫПОЛНЕНИЕ УЧЕБНЫХ ПРОЕКТОВ

Начиная работу над проектом:

1. Рассмотрите темы и цели предлагаемых учебных проектов. Выберите проект, соответствующий вашей осведомлённости, интересам, а также интересам ваших товарищей, поскольку предполагается работа в команде.
2. Определитесь с темой, обсудите в малой группе своих единомышленников ролевое распределение (возможно, вы попробуете себя в одной из профессий: редактора, дизайнера, фотокорреспондента, менеджера и т. п.); определите пути поиска ответов на проблемные вопросы в соответствии с указанной целью исследования; сформулируйте задачи для исследования проблемы; создайте план реализации проекта; спланируйте предполагаемый результат реализации проекта и способ его презентации.
3. В процессе выполнения проектных заданий нужно обсудить следующие вопросы:
   1) Каковы основные идеи вашего проекта? Какие идеи вам нужно будет исследовать и изучить дополнительно?
   2) Где вы будете искать нужные данные? Какие данные можно найти в газетах, книгах или Интернете? С какими людьми вы, возможно, захотите и сможете встретиться и обсудить проблему?
   3) Что будет результатом вашего проекта? Каким образом вы планируете провести презентацию результатов исследования?
   4) Каких ресурсов (информационных, материальных, человеческих) вам не хватает для выполнения заданий проекта и как их можно получить?

При подготовке информационных материалов для публикации лучше использовать единый стиль оформления, который можно разработать самостоятельно или воспользоваться одним из стандартных макетов редактора презентаций. Презентация может содержать созданные изображения, видеофрагменты, диаграммы, построенные на основе данных таблиц, расчёты, выполненные в электронных таблицах и т. п.

Взаимооценивание и рефлексию работы над проектом целесообразно проводить на всех этапах его реализации. Для этого составляйте и используйте контрольные списки для каждого из этапов реализации проекта, оценивайте вклад каждого в решение общей задачи, помогайте друг другу.

Обсуждайте каждую идею вместе, используйте для этого карты знаний, таблицы планирования и листы-напоминания о планах и их сроках. Привлекайте к реализации своего проекта взрослых и сверстников — их советы и взгляд со стороны могут быть вам полезны.

При оценивании своей работы в группе воспользуйтесь таким чек-листом:

- Я слушал/слушала, когда другие говорили.
- Я предлагал/предлагала свои идеи.
- Я спрашивал/спрашивала других об их идеях.
- Я делился/делилась материалами и инструментами.
- Я обращался/обращалась к моим партнёрам за помощью, когда она мне была нужна.

- Я помог/помогла кому-то в моей группе.
- Я сказал/сказала другим, что мне нравятся их идеи.
- Я придерживался/придерживалась очерёдности в высказываниях и старался/старалась, чтобы другие тоже следили за этим.

Всегда проверяйте, все ли участники группы понимают задания, прислушиваются к идеям и мнениям других, терпеливо объясняют друг другу, как можно решить проблему, делятся всеми материалами друг с другом.

При работе над проектом, в зависимости от выбранного направления, вы можете подготовить следующие материалы: список полезных информационных источников, программный проект, собственную коллекцию изображений и фото, звуковой файл и видеоролик, текстовые документы, диаграммы, таблицы, инструкции, формы для опроса, электронные таблицы с расчётами и диаграммами, а также презентацию и компьютерный фотоальбом.

Заключительным этапом проекта может стать круглый стол, где вы вместе со своими товарищами, выполняющими задания проекта, сможете представить результаты своей работы, обсудить идеи, а также убедиться в преимуществах применения современных информационно-коммуникационных технологий как для организации проектной деятельности, так и в других сферах жизни человека.

Во время подготовки к этому мероприятию вам нужно обобщить все наработки, подготовить слайдовую презентацию для сопровождения вашего выступления. Для информирования приглашённых на круглый стол гостей (учащихся других классов, учителей, родителей и др.) по тематике, программе и регламенту его проведения целесообразно создать группой текстовый документ.

Тема 1: «Информационные товары и услуги».

Проблемный вопрос: Какой уровень информационных услуг в вашем городе (районе, посёлке)?

Цель исследования: Проанализировать структуру информационных услуг, динамику их изменений; выявить, какие информационные услуги и в каком объёме можно получить в вашем регионе; проанализировать структуру рынка информационных продуктов и услуг, динамику изменений на нём; выявить наиболее значимые для учащихся интернет-ресурсы и оценить возможности удовлетворения учебных потребностей учащихся.

Тема 2: «Совместная деятельность в сети Интернет».

Проблемный вопрос: Как организовать совместную деятельность в сети Интернет?

Цель исследования: Выявить наиболее значимые для учащихся интернет-ресурсы; проанализировать их структуру и динамику изменений; выяснить возможности и потребности учащихся в использовании социальных сервисов Веб 2.0; сформулировать рекомендации по эффективному использованию Интернета для общения и совместной творческой деятельности, генерации идей, обсуждения плана их реализации и полученных результатов.

Тема 3: «Информационная безопасность».

Проблемный вопрос: Как обеспечить информационную безопасность и соблюдение авторских прав?

Цель исследования: Выяснить, кому и от кого следует защищаться при организации и проведении информационной деятельности, в частности, в сети Интернет; как обеспечить собственную информационную безопасность; определить способы защиты авторских прав в Интернете, правила корректного обращения с авторскими правами других; найти ресурсы, которые лучше использовать в учебных исследованиях, чтобы не нарушать авторские права.

Тема 4: «Бизнес-план: планирование успеха».

Проблемный вопрос: Ученики и бизнес — это реальность?

Цель исследования: Ознакомиться с правилами составления бизнес-планов; узнать, какие фонды и организации проводят конкурсы на лучший бизнес-план среди молодёжи; проанализировать потребности вашего региона (города, посёлка); создать собственный бизнес-план и написать аргументированное письмо заинтересованным лицам с просьбой принять его к рассмотрению и позволить представить собственные идеи.

Тема 5: «В здоровом теле здоровый дух».

Проблемный вопрос: Как влияет здоровый образ жизни на успешность человека?

Цель исследования: Изучить истории успешных людей разных профессий и определить критерии успешности; оценить роль здорового образа жизни как фактора успеха человека в социальной, профессиональной сферах и семейной жизни; обсудить способы организации здорового образа жизни и сформулировать рекомендации «10 правил жизненного успеха».

Тема 6: «Добро начинается с тебя».

Проблемный вопрос: Что я могу сделать для улучшения жизни уже сегодня?

Цель исследования: Изучить проблемы вашего региона: социальные, экологические, культурные; определить пути их решения силами учащихся вашего класса, школы, общества; спланировать и провести социальную акцию: благотворительный концерт или ярмарку, экологический рейд, фестиваль социальной рекламы и т. п.

Тема 7: «Электронное управление в школе».

Проблемный вопрос: Как информационные технологии могут сделать жизнь школьного сообщества лучше?

Цель исследования: Изучить информационные ресурсы, необходимые для организации самоуправления в школе: для сбора и передачи сообщений между учащимися школы, получения обратной связи об обсуждаемых идеях, для разработки бюджета коллективных дел и т. п.; определить форму представления ресурсов, способ получения и передачи данных; обсудить, какое влияние окажут предлагаемые ресурсы на мобильность, успешность, формирование лидерских качеств учащихся, достижение лучших результатов в обучении и организации досуга; спланировать и провести представление системы электронного управления.

Тема 8: «Наш умный дом».

Проблемный вопрос: Как помочь моей семье эффективно планировать и вести домашнее хозяйство?

Цель исследования: Изучить расходы семейного бюджета на ведение домашнего хозяйства; разработать информационную систему поддержки учёта, планирования, распределения, прогнозирования семейного бюджета; определить источники экономии средств и рассчитать их; сформулировать рекомендации по эффективному ведению домашнего хозяйства и предложить инструменты для автоматизации расчётов, представить их общественному совету или родительскому комитету для использования в семьях одноклассников.

ОГЛАВЛЕНИЕ