ИНФОРМАТИКА

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

11

ИНФОРМАТИКА

Учебное пособие для 11 класса
учреждений общего среднего образования
с русским языком обучения
(с электронными приложениями)

Допущено
Министерством образования
Республики Беларусь

Минск «Народная асвета» 2021

УДК 004(075.3=161.1)
ББК 32.81я721
И74

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

Рецензенты:
кафедра информационных технологий и моделирования экономических процессов учреждения образования «Белорусский государственный аграрный технический университет» (кандидат педагогических наук, доцент, заведующий кафедрой *О. Л. Сапун*); учитель информатики квалификационной категории «учитель-методист» государственного учреждения образования «Гимназия № 1 г. Бреста» *И. Ю. Горбацевич*

Электронное приложение «Информационные технологии» (базовый уровень), электронное приложение для повышенного уровня размещены на ресурсе profil.adu.by

ISBN 978-985-03-3636-1

СОДЕРЖАНИЕ

От авторов

Уважаемые одиннадцатиклассники!

Мы живем во время стремительных перемен, когда информатика становится важнейшей составляющей всей системы научного познания, определяет пути формирования информационного общества.

В современном мире необходимым качеством личности становится высокий уровень ее информационной культуры, а информация является одним из основных ресурсов, который определяет развитие страны, ее будущее.

Структурно в информатике выделяют такие разделы, как теоретическая информатика и прикладная информатика. В школьном курсе представлены оба эти раздела.

Теоретическая информатика — фундаментальная составляющая учебного предмета. Изучение общих закономерностей протекания информационных процессов, основных понятий логики, теории алгоритмов, языков программирования позволяет развивать у учащихся логическое и алгоритмическое мышление, воспитывать информационную культуру, формирует научное мировоззрение.

Прикладная информатика направлена на применение понятий и результатов теоретической информатики к решению конкретных задач в конкретных прикладных областях. Прикладная информатика включает в себя архитектуру компьютера и компьютерных сетей, компьютерную графику, компьютерное моделирование, базы данных, информационные технологии и др. Изучение прикладной информатики направлено на формирование компьютерной грамотности — владение необходимым набором знаний и навыков работы на компьютере для обработки, хранения, передачи, получения и использования данных. Прикладная информатика самая динамичная составляющая информатики.

В печатном варианте учебного пособия содержатся главы «Введение в объектно-событийное программирование», «Основы веб-конструирования» и «Информационные технологии в обществе». Глава «Компьютерное моделирование» представлена в электронном приложении к учебному пособию. Кроме того, в электронном приложении содержится альтернативный вариант для изучения темы «Введение в объектно-событийное программирование» в среде программирования Delphi, которая является профессиональной средой быстрой разработки приложений с оконным интерфейсом. Данная возможность позволит увидеть, что одна и та же задача может быть решена с использованием различных инструментов.

Материалы учебного пособия разделены на две колонки. Цвет фона поможет вам разобраться в назначении размещенной на нем информации.

— основные материалы, обязательные для изучения;

— примеры, иллюстрирующие основные материалы;

— определения основных понятий;

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

В учебном пособии используются следующие условные обозначения:

— вопросы и задания для проверки знаний;

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

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

— раздел «Упражнения» содержит задания, при выполнении которых может быть использована информация, размещенная на Национальном образовательном портале;

* — упражнение или пример для любознательных.

В тексте некоторых упражнений вам будет предложено открыть файл. Это означает, что упражнение можно выполнить, используя файл, размещенный на Национальном образовательном портале http://profil.adu.by → категория «Информатика/Інфарматыка» → курс «Информационные технологии. 11 класс (базовый уровень)» или «Информатика. 11 класс (повышенный уровень)». Зайти на сайт и скачать файлы к упражнениям можно также с помощью матричного QR-кода:

Имя файла для скачивания содержит номер параграфа и номер упражнения. Например, имя файла upr10_3 означает, что файл относится к третьему упражнению десятого параграфа.

Также на портале размещены файлы с проектами, рассмотренными в примерах. Такие файлы имеют имя pr3_1.zip (проект для примера 3.1).

Авторы учебного пособия постарались сделать так, чтобы вы смогли освоить необходимые современному человеку знания и навыки в условиях стремительного изменения информационных технологий. В электронном приложении к учебному пособию изложены те темы 11-го класса, которые относятся к прикладной информатике.

— приложение «Информационные технологии» размещено на электронном образовательном ресурсе (http://profil.adu.by).

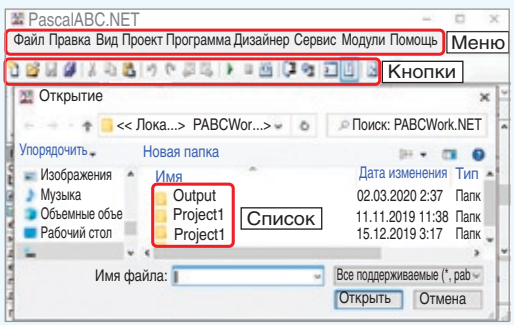
— материал для повышенного уровня размещен на электронном образовательном ресурсе (http://profil.adu.by).

Глава 1
ВВЕДЕНИЕ В ОБЪЕКТНО-СОБЫТИЙНОЕ ПРОГРАММИРОВАНИЕ

§ 1. Объектно-событийная модель работы программы

Основателем RAD считается сотрудник IBM, британский консультант по информационным технологиям Джеймс Мартин (1933—2013), который в начале 1990-х гг. сформулировал основные принципы RAD, основываясь на идеях Барри Бойма и Скотта Шульца.

Пример 1.1. После загрузки какого-либо редактора пользователь может открыть файл для редактирования. При этом он выбирает меню **Файл**, находит в списке команду **Открыть**, выбирает нужный файл, нажимает кнопку **Открыть**. Как мы видим, чтобы открыть файл, пользователь взаимодействует с такими элементами управления, как меню, список, кнопка.



1.1. Элементы управления в приложениях с графическим интерфейсом

Современные программы, с которыми сегодня работают пользователи компьютера, отличаются от тех, которые вы создавали раньше. Основное отличие — взаимодействие пользователя с программой.

Программы, которые вы создавали в 7—10-м классах, взаимодействовали с пользователем посредством текстового интерфейса (часто его называют интерфейсом командной строки). После запуска программы вы вводили данные, программа выполнялась, и вы видели результат. И ввод, и вывод данных осуществлялся в алфавитно-цифровой форме.

Операционные системы с графическим оконным интерфейсом (например, Windows) предполагают общение пользователя с программой посредством элементов управления. К элементам управления относят: кнопки, разнообразные меню, текстовые сообщения, списки и др. При работе программы пользователь выбирает какой-либо элемент управления и совершает с ним определенное действие (пример 1.1). Если такое действие для выбранного элемента было определено, то программа его выполняет, иначе выдает сообщение об ошибке.

Многие системы программирования позволяют создавать программы с оконным интерфейсом. Такие программы называют **оконными приложениями** (Windows Form Application).

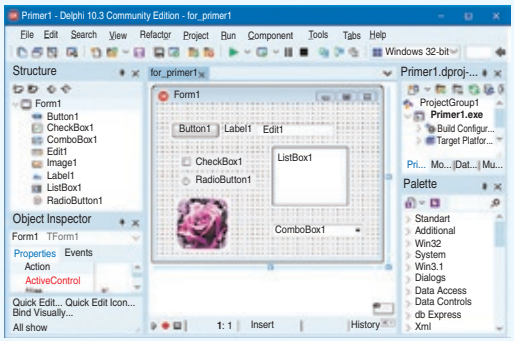
Проектирование интерфейса окна программы можно выполнять с использованием RAD-технологии (Rapid Application Development — **быстрая разработка приложений**). Технология RAD характерна для многих систем программирования. Быстрая разработка стала возможной за счет того, что элементы управления были визуализированы и собраны в специальные библиотеки — VCL (Visual Component Library — визуальная библиотека компонентов).

Различные элементы управления можно перетаскивать с палитры компонентов на форму с помощью мыши. Процесс создания интерфейса будущей программы представляется аналогом работы с неким конструктором. Программирование в RAD-средах является визуальным, поскольку код по созданию объекта не пишется, а генерируется средой. Задача программиста — написание кода по управлению готовыми компонентами.

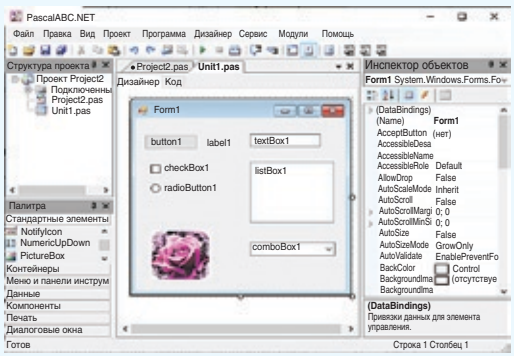
Визуальное программирование поддерживается в PascalABC и Delphi (код пишется на языке Pascal), VisualBasic, C# и др. (пример 1.2). Для обучения учащихся младших классов используется визуальное программирование в среде Скретч (Scratch).

Пример 1.2. Среды программирования, в которых реализована поддержка парадигмы визуального программирования.

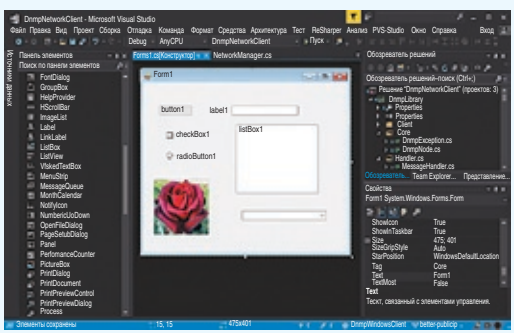
Delphi:



PascalABC:



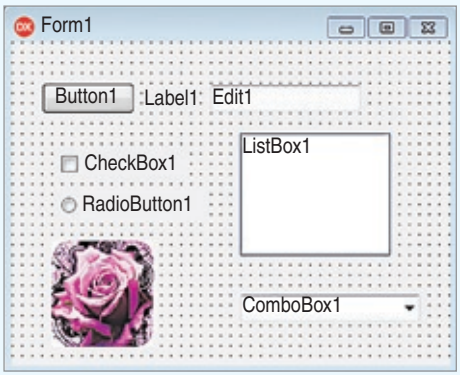
Visual Studio для языка C#:



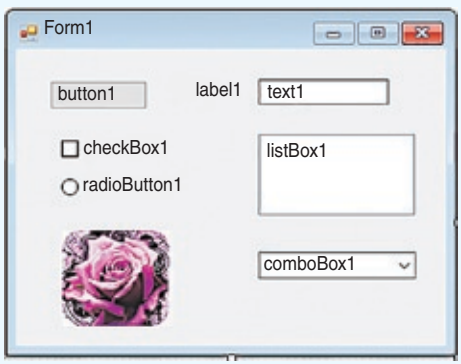
Пример 1.3. Основные элементы интерфейса:

Элемент управления	Имя
Кнопка	Button
Надпись	Label
Поле для ввода текста	TextBox (Edit)
Флажок	CheckBox
Радиокнопка	RadioButton
Список	ListBox
Выпадающий список	ComboBox
Рисунок	PictureBox (Image)

Элементы управления на форме в среде программирования Delphi:



Элементы управления на форме в среде программирования PascalABC:



Многие элементы управления в разных средах имеют одинаковые или синонимичные имена (пример 1.3).

Создаются оконные приложения как проект и состоят из нескольких файлов. Внешний вид окна будущего приложения строится на форме. Для формы сохраняются два файла — один содержит описание внешнего вида формы, другой — описание действий при выборе пользователем того или иного элемента управления. Главный файл проекта содержит описание его структуры, а также команды по созданию формы и запуску приложения.

Все элементы, размещенные на форме, и сама форма образуют систему взаимодействующих объектов. Способ их взаимодействия основан на объектно-ориентированном программировании.

Объектно-ориентированное программирование (ООП) — технология создания программ, основанная на использовании системы объектов. Каждый объект обладает набором **свойств**, которые описывают его состояние, и **методов**, характеризующих его поведение.

Объект — совокупность данных и методов работы с ними.

Организация данных внутри объекта скрыта от пользователя. Данные и способы их чтения и записи являются свойствами объекта, их можно изменять. Методы — процедуры и функции для обработки данных.

1.2. События

Организация взаимодействия между программой и пользователем управляется **событиями:** пользователь может нажать на клавишу мыши или клавиатуры, ввести текст и др.

Метод программирования, основанный на управлении событиями, называют **событийно-ориентированным программированием.**

Каждое событие связано с каким-либо объектом, которому передается управление в тот момент времени, когда происходит событие. Среди основных событий можно выделить три категории: события мыши, события клавиатуры и системные события (примеры 1.4—1.6).

Процедура (или функция), инициируемая событием, называется **обработчиком события.**

Запущенный на выполнение проект находится в ждущем режиме, реагируя на события, учтенные при его создании, вызываемые действиями пользователя или возникающими в самой программе.

Объектно-событийная модель программы предполагает следующее:

- создание объектов с присущими им свойствами и методами;
- описание событий, при которых объект может выполнять алгоритм обработки данных.

Пример 1.4. События мыши возникают в том случае, если пользователь производит какие-либо действия с мышью:

Click	Нажатие левой кнопки мыши
DblClick	Двойной щелчок левой кнопкой мыши
MouseDown	Нажатие на любую кнопку мыши. Параметры обработчика события позволяют определить, какая из кнопок была нажата и в какой точке
MouseUp	Освобождение кнопки мыши, которая была нажата
MouseMove	Перемещение указателя мыши

Пример 1.5. События клавиатуры происходят при нажатии клавиш на клавиатуре:

KeyPress	Нажатие клавиши с текстовым символом
KeyDown	Нажатие любой клавиши
KeyUp	Освобождение клавиши

Пример 1.6. Системные события управляются функциями операционной системы:

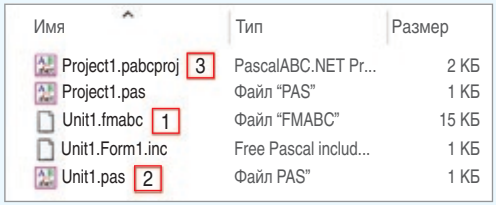
Paint	Возникает, когда элемент необходимо перерисовать
Resize	Происходит, когда размеры элемента изменяются
Enter	Возникает, когда элемент управления становится активным
Leave	Происходит, когда элемент управления перестает быть активным

? 1. Какие программы называют оконными приложениями?
2. Что понимают под событийным программированием?
3. Какие типы событий вы можете назвать?

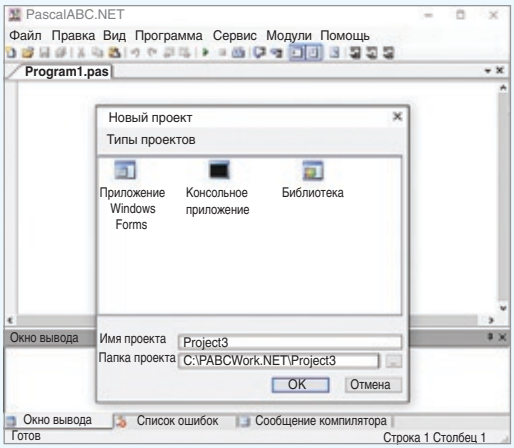
§ 2. Визуальная среда разработки программ

Работа по созданию оконных приложений рассматривается в среде программирования PascalABC.Net.

Пример 2.1. Файлы проекта:



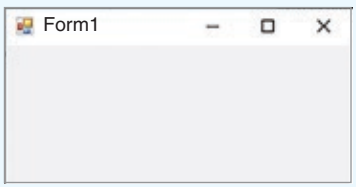
Пример 2.2. Создание проекта в PascalABC.Net:



Пример 2.3. Создание и выполнение проекта:

1. Создать папку с именем Primer1.
2. Создать проект.
3. Запустить проект на выполнение.

Вид окна приложения:



Для сохранения изменений используют команду Сохранить все (кнопка).

2.1. Структура проекта

При создании оконного приложения работают с проектом, состоящим из нескольких файлов. В разных средах программирования проект может состоять из различного количества файлов. Обязательными файлами являются следующие:

- файл формы (1), содержащий описание внешнего вида окна приложения;
- файл программного модуля (2), содержащий описание функций-обработчиков для объектов на форме;
- файл проекта (3), позволяющий связать структурные элементы проекта между собой.

(Рассмотрите пример 2.1.)

Файлы одного проекта обычно хранятся внутри отдельной папки. При компиляции приложения создается файл с расширением *.exe* и именем, совпадающим с именем проекта. Этот файл запускает работающее приложение без загрузки среды программирования. (Как скомпилировать приложение, чтобы файл с расширением *.exe* не удалялся после закрытия окна, см. в *Приложении*, с. 102).

Для создания проекта в среде PascalABC.Net нужно выполнить команды **Файл → Новый проект → Приложение Windows Form** (пример 2.2).

При создании проекта файлы сохраняются автоматически (пример 2.3).

2.2. Интерфейс среды программирования

Полное окно среды программирования PascalABC.Net при создании приложений Windows Form можно посмотреть в *Приложении* (с. 102).

Рассмотрим основные элементы.

Основное меню и **панель быстрого доступа** (пример 2.4) содержат команды для управления проектом: сохранение, загрузка, выполнение и др.

Форма (пример 2.5) служит для визуального отображения окна приложения. Во время проектирования приложения на форме отображается точечная сетка, позволяющая выравнивать помещаемые на форму компоненты.

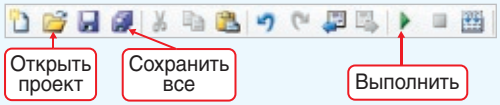
Инспектор объектов (пример 2.6) отображает свойства (или события) выбранного объекта.

В левом столбце вкладки **Свойства** перечислены все свойства объекта, которыми пользователь может управлять при проектировании приложения. В правом столбце указаны значения свойств, которые могут выбираться из списка или вводиться с клавиатуры.

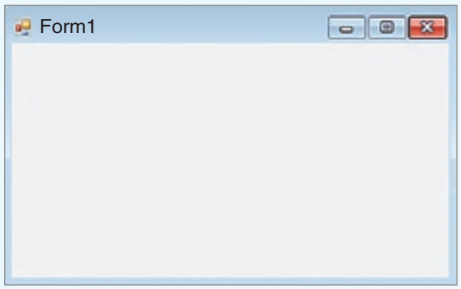
Вкладка **События** содержит список событий для объекта. Для каждого события может быть определен свой обработчик. Если обработчик для события определен, напротив события будет прописано имя процедуры (функции) обработчика.

В нижней части инспектора объектов размещено описание выбранного свойства или обработчика событий.

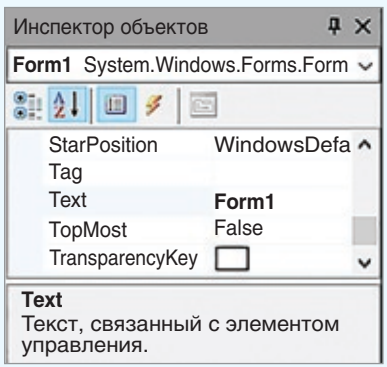
Пример 2.4. Меню и панель быстрого доступа:

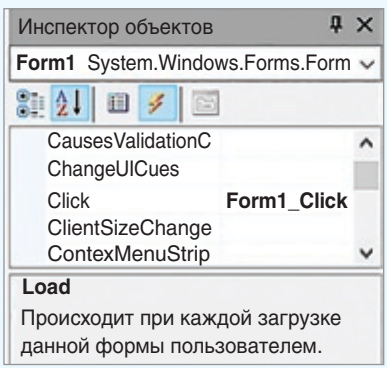


Пример 2.5. Форма:

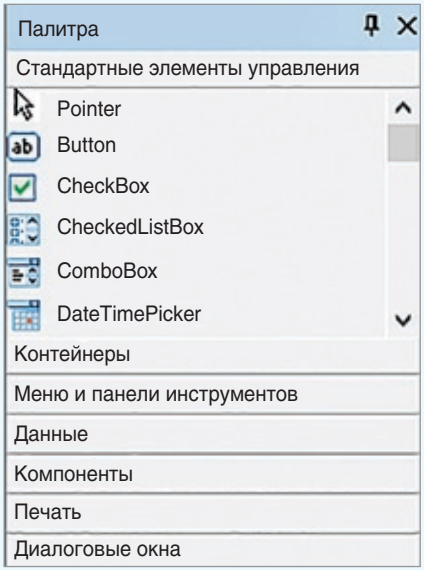


Пример 2.6. Инспектор объектов. Отображаются свойства формы:





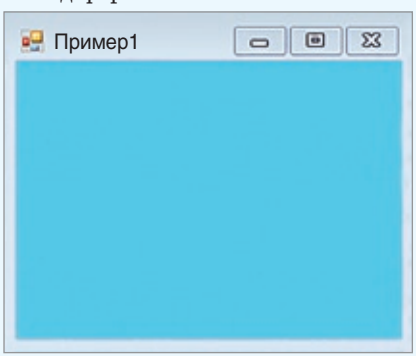
Пример 2.7. Палитра компонентов:



Пример 2.8. Изменение свойств формы в инспекторе объектов:

Свойство	Значение
Text	Пример 1
BackColor	clAqua (выбрать из списка на вкладке Интернет)
Size	250; 250
Location	200; 200

После изменения значений свойств в инспекторе объектов изменится внешний вид формы:



Палитра компонентов (пример 2.7) содержит список визуальных компонентов, объединенных в группы. Раскрытие группы происходит по щелчку с названием группы.

2.3. Работа с формой

Форма является объектом и служит для визуального отображения окна приложения. Как любой объект, форма обладает свойствами (пример 2.8).

Свойство	Назначение
Text	Заголовок формы отображается в строке заголовка окна при запуске приложения. По умолчанию — Form1
BackColor	Цвет формы. Может быть выбран один из стандартных (перечислены в списке) или задан вручную тремя числами, соответствующими RGB
Size	Высота и ширина формы. Можно указать два числа через «;» или развернуть свойство, нажав значок >, и получить возможность ввода значений Width и Height
Location	Горизонтальная и вертикальная координаты положения верхнего левого угла окна формы на экране. Можно указать два числа через «;» или развернуть свойство, нажав значок >, и получить возможность ввода значений X и Y
(Name)	Имя (внутреннее) формы. Используется в программном коде для обращения к объекту. Является идентификатором

Для создания обработчика событий формы нужно в инспекторе объектов перейти на вкладку **События** (), выбрать событие. Процедура генерируется автоматически при двойном клике мышью в пустой строке напротив выбранного события. После этого среда переключается на страницу, на которой пишется код (пример 2.9).

Имя процедуры-обработчика состоит из названия компонента, над которым происходит событие, и названия события (Form_Click).

Для каждого объекта определен обработчик по умолчанию, который создается при двойном клике по объекту. Для формы таким обработчиком будет `Form1_Load` — событие, которое происходит при загрузке формы.

Для переключения между окном программного кода и конструктором дизайна формы можно использовать вкладки **Дизайнер** и **Код** в верхней части окна приложения: Дизайнер Код.

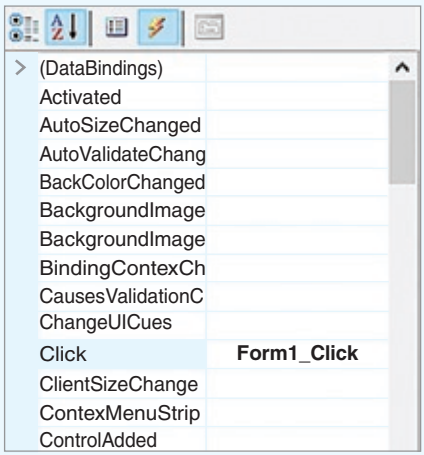
При создании процедур-обработчиков свойства объектов можно изменять программно. Для этого нужно обратиться к свойству по его имени и присвоить новое значение. Например, для изменения цвета формы нужно записать следующую команду:

```
BackColor := Color.Red;
```

Система Pascal позволяет упростить ввод сложных имен в код программы. После того как вы наберете часть сложного имени, на экране появится список со всеми свойствами и методами, которые относятся к этому объекту (пример 2.10).

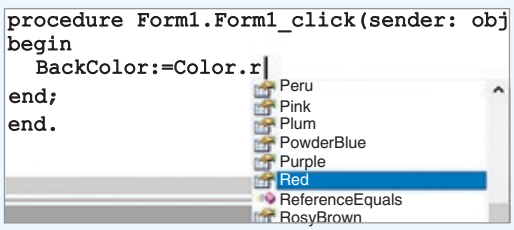
Пример 2.9. Создание обработчика события Click (клик левой клавишей мыши) для формы.

1. Выбор события в инспекторе объектов:



2. Окно программного кода со вставленным обработчиком:

Пример 2.10. Подсказка системы при вводе свойств объекта:



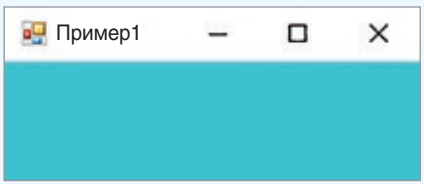
Если вы введете первые буквы названия свойства (метода), то курсор переместится в списке к тем свойствам и методам, названия которых начинаются на эти буквы. После этого нужное свойство можно вставить в программу щелчком мыши или нажатием клавиши Enter. Если список не появился, его можно вызвать комбинацией клавиш Ctrl + пробел.

Пример 2.11. Код процедуры-обработчика:

```
procedure Form1.Form1_Click
 (sender: Object; e: EventArgs);
begin
  BackColor := Color.Red;
end;
```

Чтобы убедиться в правильности работы программы, нужно запустить проект и проверить, что при выполнении щелчка мышью по форме ее цвет меняется на красный.

Форма после запуска проекта:



Форма после щелчка мыши по ней:

Пример 2.11. Создать обработчик события для щелчка левой клавишей мыши по форме, в результате которого цвет формы должен поменяться на красный (продолжить работу с примером 2.8).

Этапы выполнения задания

1. Перейти на вкладку **Events** в окне инспектора объектов.
2. Выполнить двойной щелчок в поле напротив события **OnClik**.
3. В окне редактора кода в процедуре

```
Form1.Form1_Click(sender: Object;
e: EventArgs);
```

вписать команду

```
BackColor := Color.Red;
```

Все изменения свойств формы, которые производили в примере 2.8, можно описать программно. Для этого создается обработчик события `Form1_Load`.

4. Сохранить изменения в проекте.

1. Какие элементы среды PascalABC отображаются на экране после создания проекта?
2. Какие файлы входят в состав приложения, создаваемого в PascalABC?
3. Для чего предназначена форма?
4. Для чего используют инспектор объектов?
5. Какие свойства форм вы можете назвать?
6. Как создать обработчик события?

Упражнения

1 Внесите изменения в проект из примера 2.11 так, чтобы цвет формы менялся случайно. Изменять цвет можно с помощью функции FromArgb. У этой функции четыре параметра: прозрачность (альфа-канал, интенсивность красного цвета, интенсивность зеленого цвета, интенсивность синего цвета). Генерация случайных чисел происходит следующим образом. Сначала создается переменная, являющаяся объектом класса Random (команда `var rnd: Random := new Random();`). Каждое новое случай-

ное число можно получить, обращаясь к методу `next(x)`, где `x` задает полуинтервал `[0, x)`. Команда смены цвета будет выглядеть следующим образом:

```
BackColor := Color.FromArgb(255, rnd.next(256), rnd.next(256), rnd.next(256));
```

2 Создайте проект, в котором при двойном клике мыши по форме ее размеры будут увеличиваться на 5.

1. Создайте и сохраните в новой папке проект.
2. Измените свойство Text формы на Упражнение 2.
3. Создайте обработчик события мыши `DblClick`.
4. Для изменения ширины и высоты формы можно воспользоваться командами:

```
Width := Width + 5;
Height := Height + 5;
```

5. Сохраните изменения в проекте.
6. Запустите проект и проверьте его работу.

3 Создайте проект, в котором цвет формы будет меняться при наведении на нее мыши, например с желтого на зеленый.

1. Измените свойство Text у формы на Упражнение 3.
2. Установите желтый цвет формы.
3. Создайте обработчики для двух событий мыши: `MouseEnter` и `MouseLeave`.
4. В коде события `MouseEnter` установите зеленый (Green) цвет формы, а коде события `MouseLeave` — желтый (Yellow).
5. Сохраните изменения в проекте.
6. Запустите проект и проверьте его работу.

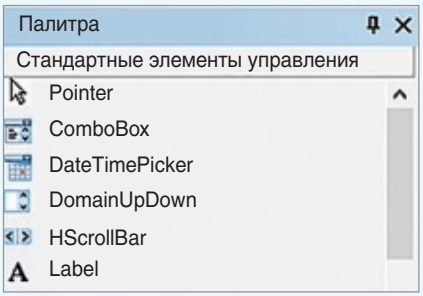
§ 3. Проектирование интерфейса оконного приложения с использованием элементов управления

3.1. Основные элементы управления

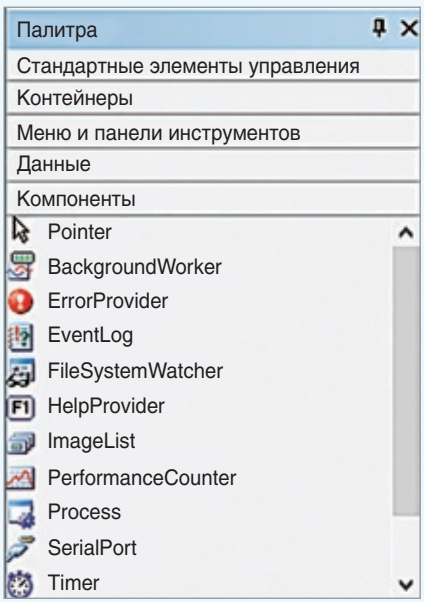
Элементами управления называются объекты, которые используются для отображения данных или организации взаимодействия между пользователем и приложением с помощью мыши или клавиатуры. Они собраны в специальные библиотеки компонентов, которые ОС использует для обеспечения единообразного интерфейса

Для элементов управления используется и другое название — виджеты. Слово употребляется примерно с 1920-х гг. в американском английском для обозначения простой, но необходимой вещи, маленького изделия. Одним из вариантов происхождения этого слова считается словослияние «window gadget» (букв. «оконное приспособление»), также произошедшее в начале XX в.

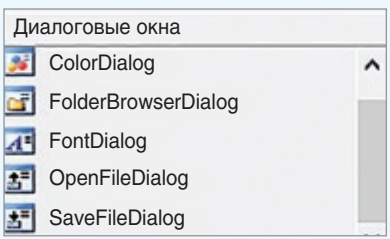
Пример 3.1. Палитра **Стандартные элементы управления:**



Пример 3.2. Палитра **Компоненты:**



Пример 3.3. Палитра **Диалоговые окна:**



прикладных программ. Наиболее распространенными элементами управления являются: кнопки, редактируемые поля, списки выбора, флажки, переключатели и т. д.

Компоненты библиотеки размещаются на различных страницах палитры компонентов. Каждая страница имеет свое название. На странице **Стандартные элементы управления** (пример 3.1) размещены наиболее употребляемые компоненты:

- Кнопка (Button)
- Надпись (Label)
- Поле для ввода текста (TextBox)
- Флажок (CheckBox)
- Радиокнопка (RadioButton)
- Список (ListBox)
- Выпадающий список (ComboBox)
- Рисунок (PictureBox)

Внешний вид этих компонентов на форме был представлен в примере 1.4.

Другие страницы палитры компонентов показаны в примерах 3.2 и 3.3.

Одним из наиболее используемых компонентов палитры **Компоненты** является компонент Таймер (Timer).

Палитра **Меню и панели инструментов** содержит компоненты, необходимые для создания главного меню программы или контекстных меню для различных объектов, помещенных на форму.

Палитры Печать и **Диалоговые окна** содержат компоненты, обеспечивающие стандартные диалоги операционной системы: открытие и сохранение файла, выбор цвета, установки параметров шрифта, настройки принтера и управление печатью.

Палитра **Данные** содержит компоненты для работы с таблицами баз данных.

3.2. Элемент управления кнопка (Button)

Компонент **кнопка** относится к элементам управления. На панели компонентов **Стандартные элементы управления** кнопка изображена в виде ab Button, имя объекта — button. Кнопка, помещенная на форму, получает имя buttonN, где N — номер 1, 2, 3... (пример 3.4). При необходимости кнопку можно переместить в любое место формы. Ключевые точки позволят установить нужный размер кнопки.

Некоторые свойства кнопки перечислены в таблице (пример 3.5).

Как видно из таблицы, многие свойства кнопки совпадают по именам и назначениям со свойствами формы, поэтому в дальнейшем для компонентов будут указываться только те свойства, которые отличны от уже описанных для других компонентов.

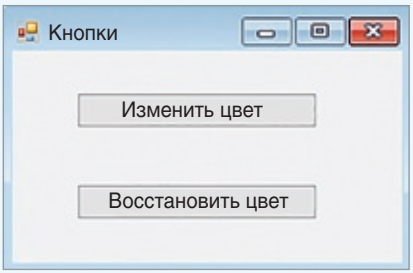
Основным событием кнопки является Click. Для создания обработчика события Click для кнопки можно поступить так же, как и при создании аналогичного обработчика для формы: выбрать событие на вкладке **События** (⚡) и выполнить двойной щелчок в поле напротив события Click. Можно просто выполнить двойной щелчок по кнопке. (Для формы основным событием является событие Load, поэтому при двойном щелчке по форме создается обработчик события Load.)

Пример 3.4. Компонент *кнопка* на форме:

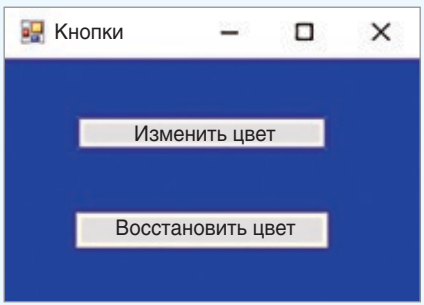
Пример 3.5. Свойства кнопки:

Некоторые свойства кнопки Button	
Свойство	**Назначение**
Text	Заголовок (внешнее имя) кнопки, текст, который отображается на кнопке. По умолчанию — button1
Font	Свойства шрифта для подписи заголовка. Свойство Font является сложным, о чем свидетельствует значок >, при нажатии на который раскрываются все свойства шрифта
Height	Высота кнопки
Weight	Ширина кнопки
Left	Горизонтальная координата положения верхнего левого угла кнопки на форме
Top	Вертикальная координата положения верхнего левого угла кнопки на форме
(Name)	Имя (внутреннее) кнопки. Используется в программном коде для обращения к объекту. Является идентификатором
Enabled	Значение True этого свойства обеспечивает доступность кнопки для мыши или клавиатуры
Visible	Значение True этого свойства обеспечивает видимость кнопки во время выполнения приложения

Пример 3.6. Внешний вид формы в режиме конструктора дизайна:



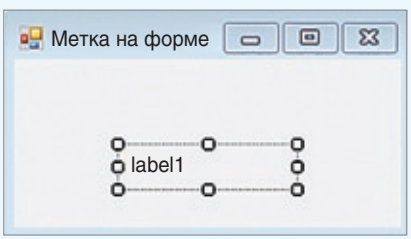
Внешний вид формы при выполнении:



Свойство кнопки `BackColor` позволяет менять ее цвет. Для обращения к этому свойству в программе используют запись: `button1.BackColor`.

Свойство кнопки BackgroundImage позволяет вставить на кнопку изображение, хранящееся в графическом файле. Для вставки можно использовать кнопку ..., Далее выбрать файл с рисунком.

Пример 3.7. Компонент *метка* на форме:



Пример 3.6. Создать проект, разместив на форме две кнопки. При нажатии на одну из них цвет формы должен измениться на синий, а при нажатии на вторую — должен восстановиться исходный цвет.

Этапы выполнения задания

1. Создать на форме две кнопки.
2. Изменить свойство Text у кнопки button1 на **Изменить цвет.**
3. Изменить свойство Text у кнопки button2 на **Восстановить цвет.**
4. Создать обработчик события Click для кнопки button1 и изменить цвет формы. Команда

```
BackColor := Color.Blue;
```

5. Создать обработчик события Click для кнопки button2 и изменить цвет формы на первоначальный (название цвета формы указано в поле Color инспектора объектов). Команда

```
BackColor := SystemColors.Control;
```

6. Сохранить изменения в проекте.

Название цвета `SystemColors.Control` задает не какой-то определенный цвет. Это цвет элемента управления, заданный цветовой схемой Windows. Поэтому он не обязательно будет серым.

3.3. Элемент управления *метка* (Label)

Компонент **метка** предназначен для отображения текста на форме. На панели компонентов **Стандартные элементы управления** метка изображена в виде A Label, имя объекта — label. Кнопка, помещенная на форму, получает имя labelN, где N — номер 1, 2, 3... (пример 3.7).

Некоторые свойства метки, отличные от свойств кнопки, перечислены в таблице (пример 3.8). Основным событием для метки является Click.

Пример 3.9. Создать проект, в котором описана возможность выполнять следующие действия: после запуска программы в окне с именем «Работаем с кнопкой и меткой» при щелчке мыши по кнопке «Приветствие» появляется сообщение «Здравствуй, мир!».

Этапы выполнения задания

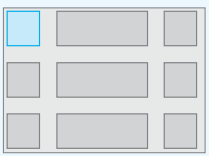
1. Изменить свойство Text формы на «Работаем с кнопкой и меткой».
2. Добавить на форму кнопку button1.
3. Изменить свойство Text кнопки на «Приветствие».
4. Добавить на форму метку label1.
5. Изменить свойства шрифта для компонента label1. Нажать кнопку ... в поле Font (цвет шрифта — синий, размер — 20, стиль — жирный курсив).
6. Очистить поле Text у метки.
7. Установить значение true у свойства метки Autosize.
8. В обработчик события Click для кнопки button1 вписать команду

```
label1.Text := 'Здравствуй, мир!';
```

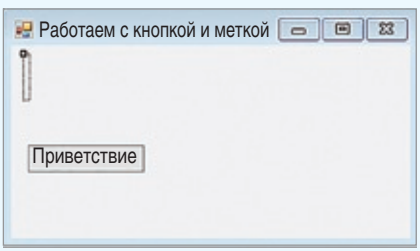
3.4. Элемент управления *текстовое* поле (Edit)

Текстовое поле — компонент, который предназначен для ввода и вывода текстовой информации. На панели компонентов **Стандартные элементы управления** текстовое поле изображено в виде abl TextBox, имя объекта — TextBox.

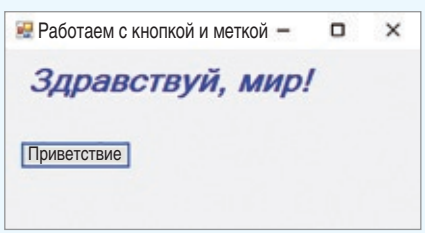
Пример 3.8. Свойства метки:

Свойство	Назначение
Text	Отображает введенный текст на форме
BackColor	Устанавливает цвет фона метки, который по умолчанию совпадает с цветом формы. Фон метки можно сделать прозрачным, задав свойству BackColor значение Color.Transparent
AutoSize	Значение true этого свойства приводит к автоматическому изменению размеров метки в соответствии с длиной текста
TextAlign	Выравнивание текста относительно границ метки: 

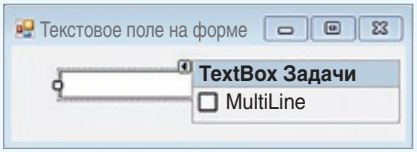
Пример 3.9. Форма на этапе конструирования:



Работающее приложение:



Пример 3.10. Компонент *текстовое поле* на форме:



Пример 3.11. Свойства текстового поля:

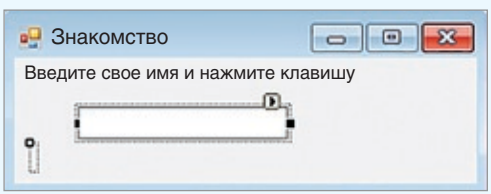
Свойство	Назначение
BorderStyle	Определяет границу вокруг текстового поля: None (нет границы), FixedSingl или Fixed3d (есть граница)
MaxLength	Ограничивает количество символов, которые можно ввести в TextBox
ReadOnly	Значение true запрещает редактирование текста, отображаемого в TextBox
Text	Содержит вводимый или выводимый текст

Текстовое поле часто называют текстовым редактором, поскольку оно снабжено такими функциями, как:

- копирование выделенного текста в буфер обмена (Ctrl+C);
- вырезание выделенного текста в буфер обмена (Ctrl+X);
- вставка текста из буфера обмена в позицию курсора (Ctrl+V);
- отмена последней команды редактирования (Ctrl+Z).

Свойство Multiline определяет, каким будет редактор — однострочным или многострочным.

Пример 3.12. Форма на этапе конструирования:



Текстовое поле, помещенное на форму, получает имя TextBoxN, где N — номер 1, 2, 3... (пример 3.10).

В отличие от ранее рассмотренных компонентов, свойство Text у текстового поля по умолчанию пусто (у других — совпадает с именем компонента). Некоторые свойства компонента TextBox приведены в таблице (пример 3.11).

Значение свойства Text компонента *текстовое поле* может изменяться программно или при вводе с клавиатуры. Основным событием для TextBox является TextChanged, которое происходит при изменении компонента. Наиболее часто программируют событие KeyPress, которое позволяет определить, какая клавиша была нажата.

Пример 3.12. Создать проект, в котором пользователя попросят ввести его имя, а затем, после нажатия клавиши Enter, будет выдано сообщение «Имя, приятно с Вами познакомиться!»

Этапы выполнения задания

1. Изменить свойство Text у формы на «Знакомство».
2. Разместить на форме две метки и текстовое поле.
3. Изменить свойство Text у label1 на «Введите свое имя и нажмите клавишу Enter».
4. Очистить поле свойства Text у Label2.
5. Написать обработчик события KeyPress для компонента Edit1, который будет проверять нажатие клавиши ввода (код клавиши Enter — 13). Если клавиша нажата, то поменять свойство Text у label2:

```
if  e.KeyChar = #13 then
  label2.Text := TextBox1.Text +
  ', приятно с Вами познакомиться!';
```

Текстовое поле TextBox используется также для ввода и вывода чисел. При этом необходимо использовать функции для преобразования строк в числа и чисел в строки. Эти функции приведены в таблице:

Название функции	Действие
Ввод с помощью Edit	
StrToInt	Преобразование строки в целое число
StrToFloat	Преобразование строки в значение с плавающей запятой
Вывод с помощью Edit	
IntToStr	Преобразование целого числа в строку
FloatToStr	Преобразование вещественного числа в строку

В русскоязычной версии Windows в качестве разделителя целой и дробной части числа по умолчанию используется запятая. Если при вводе чисел в текстовые поля использовать точку, то будет возникать ошибка преобразования типов.

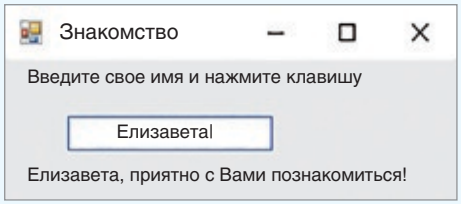
Пример 3.13. Создать проект, в котором пользователь сможет ввести число, получить его значение в квадрате и квадратный корень из этого числа.

Этапы выполнения задания

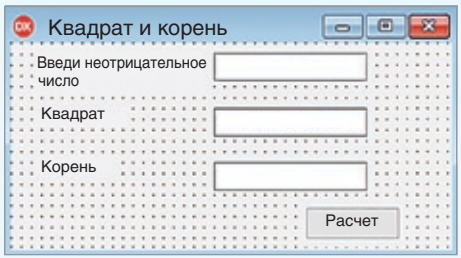
1. Изменить свойство Text у формы на «Квадрат и корень».

2. Разместить на форме три метки, три текстовых поля и кнопку.

Пример 3.12. *Продолжение.* Работающее приложение:



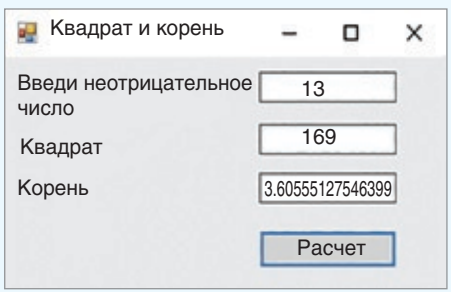
Пример 3.13. Форма на этапе конструирования:



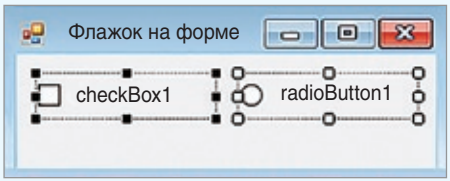
Обработчик события OnCkick для Button1:

```
procedure Form1.button1_Click
 (sender: Object; e: EventArgs);
var a, b: integer;
   c: real;
begin
  a := StrToInt(TextBox1.Text);
  b := a * a;
  c := sqrt(a);
  TextBox2.Text := IntToStr(b);
  TextBox3.Text := FloatToStr(c);
end;
```

Работающее приложение:



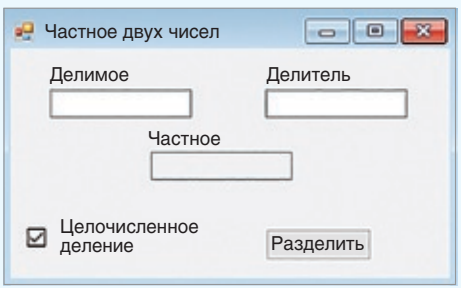
Пример 3.14. Компоненты *флажок* и *радиокнопка* на форме:



Пример 3.15. Свойства компонента *флажок*:

Свойство	Назначение
Checked	Значение true у этого свойства показывает, что установлена галочка — ☑, пустое окно индикатора — ☐ соответствует значению false
Text	Надпись возле компонента checkBox
RightToLeftt	Определяет, с какой стороны кнопки размещается надпись: Yes (надпись слева), No (надпись справа)
CheckState	Определяет состояние компонента: Unchecked — не выделен, Checked — выделен, Indeterminate (■) — промежуточное состояние. Первые два состояния соответствуют свойству Checked

Пример 3.16. Форма на этапе конструирования:



3. Изменить свойство Text у Label1 на «Введите неотрицательное число».
4. Изменить свойство Text у Label2 на «Квадрат».
5. Изменить свойство Text у Label3 на «Корень».
6. Изменить свойство Text у Button1 на «Расчет».
7. Написать обработчик Click для кнопки.

3.5*. Элементы управления *флажок* (CheckBox) и *переключатель* (RadioButton)

Флажок используется в приложениях для включения или выключения каких-либо опций. На панели компонентов **Стандартные элементы управления** флажок изображен в виде ☑ CheckBox, имя объекта — CheckBox. Флажок, помещенный на форму, получает имя checkBoxN, где N — номер 1, 2, 3... (пример 3.14). Некоторые свойства компонента checkBox приведены в таблице (пример 3.15).

Аналогичным образом используется компонент **Переключатель (радиокнопка)**. На панели компонентов Standard радиокнопка изображена в виде ◉ RadioButton, имя объекта — RadioButton. Переключатель, помещенный на форму, получает имя radioButtonN, где N — номер 1, 2, 3... (пример 3.14).

Обычно радиокнопки образуют группы взаимосвязанных индикаторов, позволяющих выбрать только одну из нескольких взаимоисключающих альтернатив. При размещении на форме нескольких переключате-

лей включенным должен быть только один из них (контейнер GroupBox).

Пример 3.16. Создать проект для вычисления частного от деления одного целого числа на другое. Числа задаются в текстовых полях. Результат вычисляется при нажатии на кнопку «Частное» и помещается в третье текстовое поле. Результат зависит от состояния флажка.

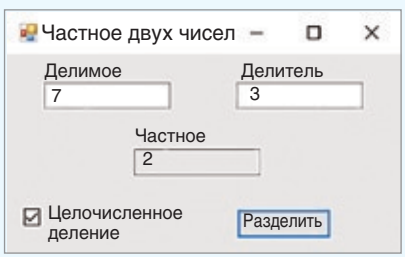
Этапы выполнения задания

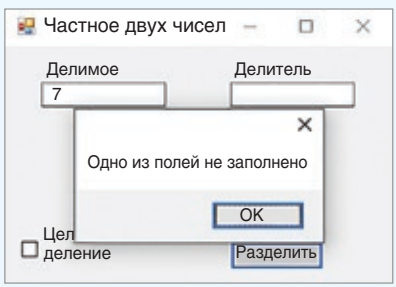
1. Поместить на форму текстовые поля (3), надписи (3), флажок и кнопку.
2. Для компонента textBox3 установить значение true для свойства ReadOnly.
3. Изменить свойство Text у компонентов label («Делимое», «Делитель», «Частное»).
4. Изменить свойство Text компонента button1 на «Разделить».
5. Изменить свойство Text компонента checkBox1 на «Целочисленное деление».
6. Написать обработчик события Click для компонента button1.
   - 6.1. Проверить, что поля компонентов textBox1 и textBox2 не пусты. Иначе вывести сообщение «Одно из полей не заполнено».
   - 6.2. Проверить состояние переключателя checkBox. Если он включен, то выполнить целочисленное деление, иначе обычное деление.
   - 6.3. Вывести результат.
7. Выполнить программу для различных значений. Проверить работу приложения, когда одно из полей textBox1 или textBox2 (или оба поля) пустые.

Обработчик события OnCkick для Button1:

```
procedure Form1.button1_Click
 (sender: Object; e: EventArgs);
var a, b, c : integer;
    d : real;
begin
  if (TextBox1.Text <> '') and
     (TextBox2.Text <> '') then
  begin
    a := StrToInt(TextBox1.Text);
    b := StrToInt(TextBox2.Text);
    if CheckBox1.checked then
    begin
      c := a div b;
      TextBox3.Text := IntToStr(c);
    end
    else
    begin
      d := a / b;
      TextBox3.Text := FloatToStr(d);
    end;
  end
  else
    MessageBox.Show('Одно из
       полей не заполнено');
end;
```

Работающее приложение:





1. Какие компоненты относят к элементам управления?
2. Как поместить компонент на форму?
3. Какие свойства компонента button вы можете назвать?
4. Какое событие является основным для компонента button?
5. Для чего предназначен компонент label?
6. В каких случаях используется компонент TextBox?
7. Для чего предназначены компоненты checkBox и radioButton?

Упражнения

1 Откройте проект из примера 3.9 и дополните его кнопкой «Очистить»[1]. Кнопка «Очистить» должна очищать текст метки (Свойству Caption присвоить значение пустой строки: ""). Сделайте случайным выбор цвета и размера шрифта у метки.

2 Откройте проект из примера 3.12 и добавьте на форму три метки и две кнопки.

1. Измените свойства компонентов в соответствии с указаниями в таблице.

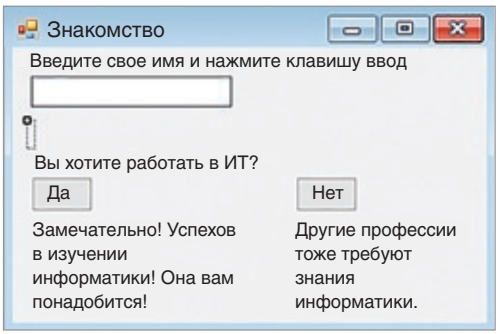
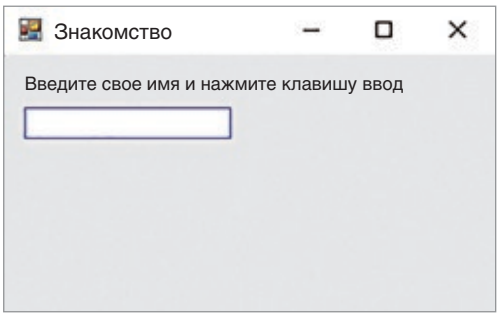
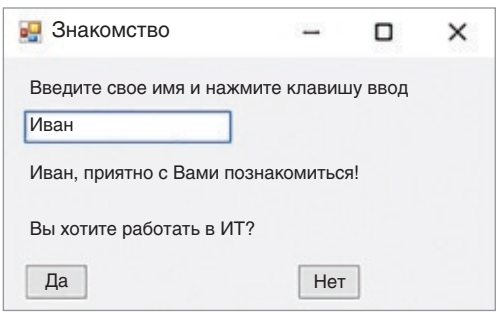
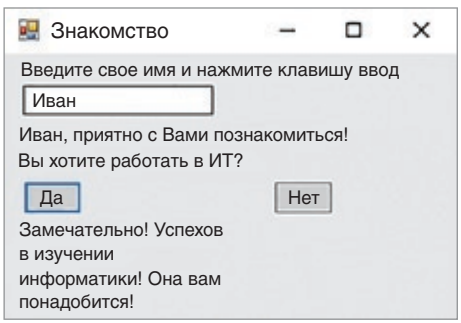
Компонент	Свойство	Значение свойства
button1	Text	Да
button1	Visible	False
button2	Text	Нет
button2	Visible	False
label3	Text	Вы хотите работать в ИТ?
label3	Visible	False
label4	Text	Замечательно! Успехов в изучении информатики! Она Вам понадобится!
label4	Visible	False
label5	Text	Другие профессии тоже требуют знания информатики
label5	Visible	False

2. Добавьте в обработчик события KeyPress команду, которая делает надпись Label3 и кнопки видимыми.

[1] Перед изменением скопируйте проект в новую папку

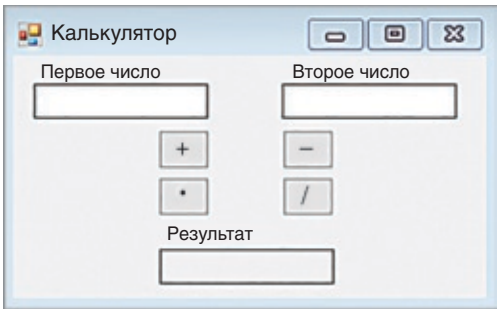
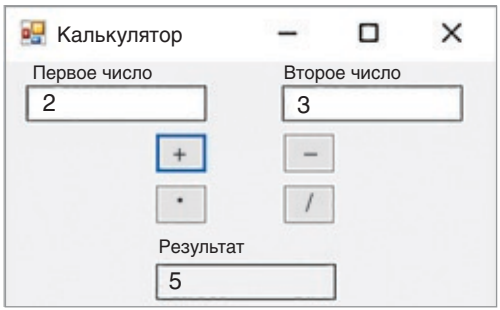
```
procedure Form1.textBox1_KeyPress(sender: Object;
e: KeyPressEventArgs);
begin
  if e.KeyChar = #13 then
  begin
    label2.Text := TextBox1.Text + ', приятно с Вами познакомиться!';
    Label3.Visible := True;
    Button1.Visible := True;
    Button2.Visible := True;
  end;
end;
```

3. Напишите обработчики Click для кнопок Button1 и Button2. Сделайте видимыми соответствующие надписи.

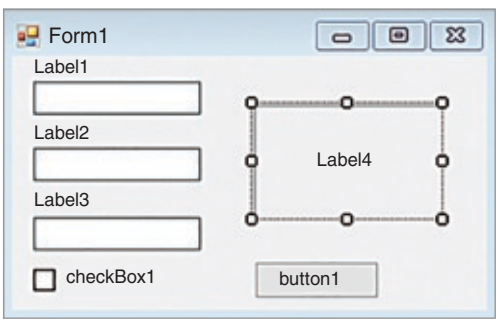
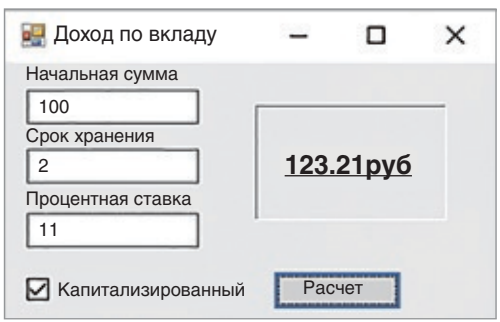
Форма на этапе проектирования	Работающее приложение после запуска
	
Работающее приложение до ответа на вопрос	**Работающее приложение после ответа на вопрос**
	

4*. Добавьте в приложение еще один вопрос. Форму ответа выберите самостоятельно.

3 Создайте проект **Калькулятор**. Разместите на форме три поля TextBox и три надписи: «Первое число», «Второе число», «Результат». Добавить кнопки для вычисления суммы, разности, произведения и частного. Запретите редактирование в поле с ответом. *Добавьте проверку деления на нуль.

Форма на этапе проектирования	Работающее приложение
	

4 Создайте проект, в котором вычисляется доход по вкладу. Программа должна обеспечивать расчет денежных сумм для простых или капитализированных вкладов. Если вклад простой, то процентная ставка начисляется от исходной суммы, и каждый месяц она одинаковая, а если капитализированный, то процентная ставка начисляется каждый месяц от суммы вклада в предыдущем месяце.

Форма на этапе проектирования	Работающее приложение
	

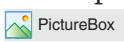
Проверьте, заполнены ли поля с исходными данными. Если нет, то выведите соответствующее сообщение.

5 Реализовать «убегающую кнопку», т. е. при наведении указателя мыши на кнопку она должна случайным образом поменять место.

6 Добавить в упражнение 5 кнопку «Домой», которая должна передвинуть «убегающую» в верхний левый угол формы.

§ 4. Элементы управления для работы с графикой

4.1. Элемент управления для вставки рисунка (PictureBox)

При создании приложений нередко возникает необходимость украсить их графическим изображением. В этом случае можно воспользоваться компонентом **изображение**. На панели компонентов **Стандартные элементы управления** компонент изображение представлен в виде 

, имя объекта PictureBox. Компонент PictureBox, помещенный на форму, получает имя PictureBox N, где N — номер 1, 2, 3... (пример 4.1).

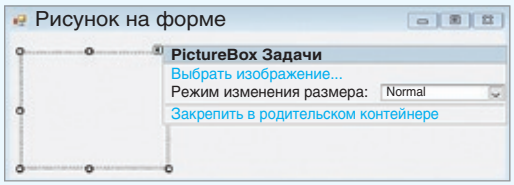
Компонент представляет собой контейнер, в который помещается изображение. Некоторые свойства компонента PictureBox приведены в таблице (пример 4.2).

Используя свойство Image, можно выбрать и загрузить изображение на этапе проектирования приложения. Изображение может быть выбрано в контейнере при нажатии на кнопку  в правом верхнем углу компонента. В этом случае рисунок сохраняется в файле формы и для работы приложения отдельного файла с рисунком не требуется.

Компонент поддерживает вставку рисунков в форматах JPEG, PNG, BMP, GIF. Если требуется обработка изображения (любые изменения рисунка), то рисунок должен быть сохранен в формате BMP. Для рисунков формата PNG или GIF с прозрачным фоном при загрузке сохраняется прозрачность.

Рисунок можно загрузить как фон формы. Для этого предназначено свойство формы BackgroundImage.

Пример 4.1. Компонент *изображение* на форме:

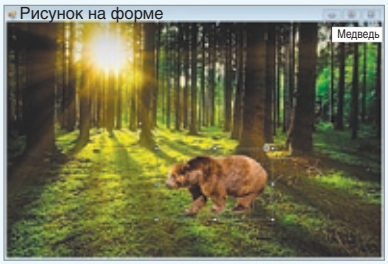


Пример 4.2. Некоторые свойства компонента *изображение*:

Свойство	Назначение
Image	Используется для отображения изображений
SizeMode	Определяет способ отображения рисунка: *Normal* — левый верхний угол рисунка совмещен с левым верхним углом контейнера; *StretchImage* — рисунок вписывается в контейнер; *AutoSize* — размер подгоняется под размер рисунка; *CenterImage* — рисунок будет отцентрирован относительно компонента; *Zoom* — при изменении размеров контейнера будут сохраняться пропорции рисунка
Margin	Определяет промежуток между полями изображения и полями другого компонента (значения All, Left, Top, Right, Bottom)
BackColor	Цвет фона изображения. Может быть прозрачным

Свойство Image компонента PictureBox обладает методом Save, который используется для сохранения изображения. Метод Load компонента PictureBox может быть использован для загрузки изображения при открытии приложения. В этом случае файл с рисунком должен находиться в папке проекта (или нужно прописать полный путь к файлу).

Пример 4.3. Форма на этапе конструирования:

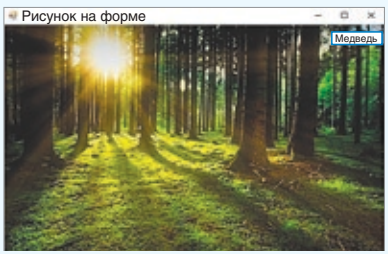


Обработчик события Ckick для Button1:

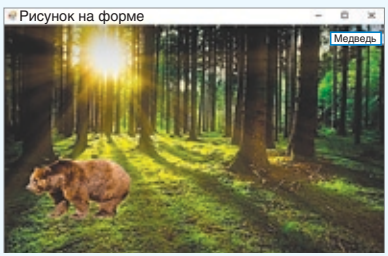
```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
begin
  PictureBox2.Top := 220;
  var rnd: Random := new Random();
  PictureBox2.Left := rnd.
Next(300);
  PictureBox2.Visible := True;
end;
```

Работающее приложение

До нажатия на кнопку:



После нажатия на кнопку:



Поскольку горизонтальное положение медведя задается случайным образом, то при каждом нажатии на кнопку медведь будет прорисован в новом месте.

Пример 4.3. Создать проект, разместить в нем фоновое изображение на форме. При нажатии на кнопку поверх фонового изображения должно появиться другое изображение.

Этапы выполнения задания

1. Установить размеры формы Height = 450, Width = 670.

2. Загрузить фоновое изображение для формы. Задать для свойства формы BackgroundImageLayout значение Stretch.

3. Поместить на форму компонент *изображение* и кнопку.

4. Для компонента PictureBox установить значение для свойства Visible = = False (изображение невидимо при запуске приложения). Размеры Height = 120, Width = 200. Свойство SizeMode = = StretchImage.

5. Загрузить изображение в компонент PictureBox. Изображение может быть формата PNG, или GIF с прозрачным фоном, или формата BMP с фоном однородного цвета. Свойство BackColor = Transparent.

6. Написать обработчик события OnClick для компонента Button1.

Если при запуске приложения изображение мерцает, то устранить мерцание можно с помощью включения двойной буферизации:

```
DoubleBuffered := true;
```

Эта команда должна быть прописана в обработчике события Load для формы.

4.2. Построение графиков функций

Пространство имен System.Drawing обеспечивает доступ к функциональным возможностям графического интерфейса Windows. Класс Graphics

предоставляет методы для рисования графических примитивов.

Основное событие для построения изображений — Paint.

Если возникает необходимость рисовать по точкам, то для этого используется класс Bitmap (точечное изображение). Каждая точка имеет координаты *X* и *Y*. Система координат такая же, как и для графического окна PascalABC.Net — точка с координатами (0, 0) расположена в верхнем левом углу, ось *OY* направлена вниз. Каждая точка имеет координаты *X* и *Y*. Координаты измеряются в пикселях. Важнейшее свойство пикселя — его цвет. Для задания цвета в PascalABC.Net можно воспользоваться несколькими способами (пример 4.4). Точка изображается с помощью команды `SetPixel(x1, y1, Color);`

Класс Graphics содержит большое количество свойств и методов, позволяющих строить изображения. Многие из методов Graphics совпадают с процедурами, которые использовались в библиотеке GraphABC среды программирования PascalABC.Net. Описание этих методов приведено в приложении.

Пример 4.5. Создать проект и построить график функции $y = x\sin x$ на промежутке, заданном пользователем.

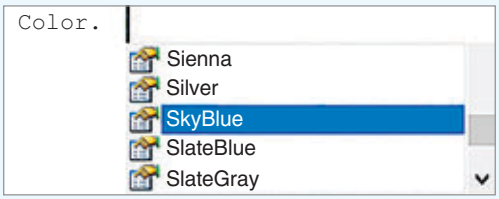
Этапы выполнения задания

1. Поместить на форму компоненты: PictureBox, два компонента Label, два компонента TextBox и компонент Button.

2. Изменить свойства Text у компонентов Label1, Label2 на x0 и xn соответственно.

Пример 4.4. Способы задания цвета в PascalABC.Net:

1. Задание цвета с помощью констант. Константы хранят имя цвета (например, Color.SkyBlue — небесно-синий, Color.Red — красный). Возможные значения появляются в выпадающем списке после ввода в программу ключевого слова Color.

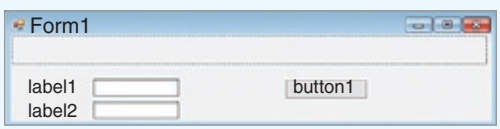


2. Для задания цвета можно воспользоваться функцией `Color.FromArgb(A, R, G, B)`, параметры которой задают прозрачность (альфа-канал) и интенсивность красного, синего и зеленого цветов соответственно. Значения параметров могут изменяться от 0 до 255.

3. Задание цвета с помощью шестнадцатеричных чисел. Пары цифр шестнадцатеричного числа задают прозрачность и интенсивность красного, зеленого и синего цветов соответственно. Полученное шестнадцатеричное число записывается как параметр функции `FromArgb`. Например, `Color.FromArgb($FF0000FF)` — синий цвет. Значения для других цветов: `$FFFF0000` — красный, `$FF00FF00` — зеленый, `$FF000000` — черный, `$FFFFFFFF` — белый.

Шестнадцатеричные числа можно перевести в десятичную систему счисления и пользоваться этими значениями. Например, `Color.FromArgb(4278190335)` — синий цвет.

Пример 4.5. Форма на этапе конструирования:



Пример 4.5. *Продолжение.*

Обработчик события Click для компонента Button1:

```
procedure Form1.button1_Click
 (sender: Object; e: EventArgs);
var x, y, h, k, x0, xn :real;
  x1, y1, n, c_x, c_y : integer;
  gr: Graphics;
  bm: Bitmap;
  p_c: Pen;
begin
//подготовка графической области
//для рисования по пикселям
 bm := new Bitmap (PictureBox1.Width,
       Picturebox1.Height);
 pictureBox1.Image := (Image)(bm);
 gr := Graphics.FromImage
      (pictureBox1.Image);
 //закраска графической области белым
 //цветом
 gr.Clear(Color.White);
 // количество точек
 n := 10000;
 // концы промежутка
 x0 := StrToFloat(TextBox1.Text);
 xn := StrToFloat(TextBox2.Text);
 // центр области построения
 c_x := PictureBox1.Width div 2;
 c_y := PictureBox1.Height div 2;
 // масштабный коэффициент
 k := PictureBox1.Width / (xn - x0);
 // шаг
 h := (xn - x0) / n;
 x := x0;
 //оси
 p_c := new Pen(Color.Black, 1);
 gr.DrawLine(p_c, 0, c_y, 2*c_x, c_y);
 gr.DrawLine(p_c, c_x, 0, c_x, 2*c_y);
 for var i := 1 to n do
 begin
  y := x * sin(x);
  x1 := trunc(x * k) + c_x;
  y1 := trunc(-y * k) + c_y ;
  //проверка попадания точки
  //в графическую область
  if (y1 >= 0) and (y1 < 2*c_y) then
   bm.SetPixel(x1, y1, Color.Blue);
  x := x + h;
 end;
end;
```

3. Изменить свойства Text у компонентов Edit1 и Edit2 на –20 и 20 соответственно.

4. Изменить свойства Text у компонента Button1 на «Построить график».

5. Написать обработчик события Click для компонента Button1 и строить в нем график функции по точкам.

5.1. Нарисовать оси координат в виде двух перпендикулярных линий, пересекающихся в центре компонента PictureBox.

5.2. Чтобы получить видимость сплошной линии, количество точек, которые образуют график функции, должно быть не менее 10 000 (n = 10 000).

5.3. Шаг изменения значения x определяется как $h = \frac{x_n - x_0}{n}$.

5.4. При построении нужно учитывать масштаб: ширина компонента PictureBox должна соответствовать длине заданного промежутка. Тогда масштабный коэффициент можно рассчитать по формуле

$$k = \frac{PictureBox1.Width}{x_n - x_0}.$$

5.5. Поскольку расположение осей координат на экране не совпадает с расположением осей, принятым в математике, то нужно преобразовать координаты: точке (0; 0) должна соответствовать точка в центре компонента PictureBox. Для этого полученное значение x нужно увеличить на величину c_x = PictureBox1.*Width* div 2, а значение y на c_y = PictureBox1.*Height* div 2. Так как ось Y направлена вниз, а не вверх, то у значения Y нужно еще поме-

нять знак на противоположный. На канве будет закрашиваться точка с координатами ($x_{ekr} = x \cdot k + c_x$, $y_{ekr} = -y \cdot k + c_y$).

5.6. Необходимо учитывать, что при вычислении значения x и y будут вещественными, а значения графических координат могут быть только целыми. Поэтому перед прорисовкой точки нужно преобразовать вещественные числа в целые с помощью функции *trunc*.

5.7. Некоторые точки графика при построении могут оказаться за пределами графической области, поэтому необходима проверка значения *y*1: значение должно быть неотрицательным и меньше высоты графической области.

4.3. Построение диаграмм

Основные принципы построения гистограмм и круговых диаграмм разбирались в 10-м классе (примеры 4.6 и 6.8). Используя аналогичные методы Graphics, можно построить диаграммы в оконных приложениях, созданных в PascalABC.Net.

Пример 4.6. Создать проект и построить гистограмму по данным массива из n элементов ($n = 10$). Описать массив с константными данными или добавить данные в массив случайным образом.

Этапы выполнения задания

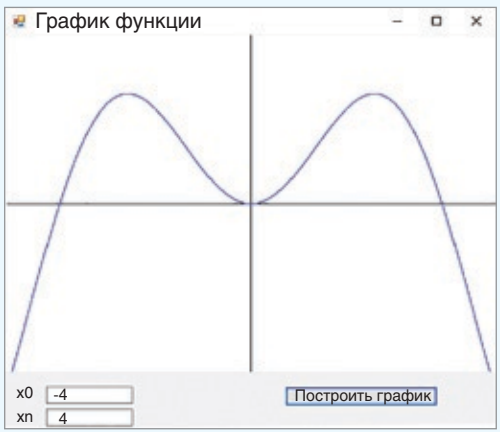
1. Поместить на форму компоненты: PictureBox и два компонента Button.

2. Изменить свойства Text у компонента Button1 на «Диаграмма с константными данными».

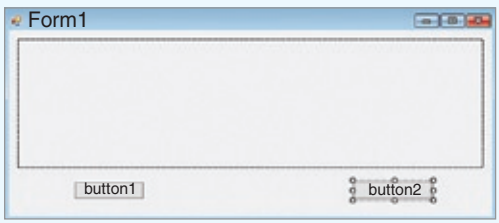
Пример 4.5. *Продолжение.*

Работающее приложение:

Изменение начальных значений концов промежутка:



Пример 4.6. Форма на этапе конструирования:

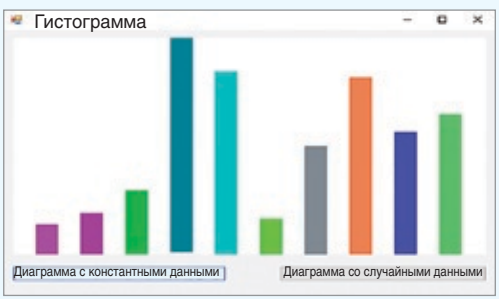


Пример 4.6. *Продолжение.*

Обработчик события Click для компонента Button1:

```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
 const a: array[1..10] of integer =
(10,14,22,75,63,12,37,61,42,48);
 n = 10;
var max, x, y1, y2, h, i, cr, cg,
  cb : integer;
  m : real;
  gr : Graphics;
  rnd : Random;
  sb : SolidBrush;
begin
 max := a[1];
 for i := 2 to n do
  if a[i] > max then
   max := a[i];
 h:= trunc(PictureBox1.Width/(2*n+1));
 m:= PictureBox1.Height / max;
 x:= h;
//подготовка графической области
//для рисования примитивов
gr := PictureBox1.CreateGraphics;
 gr.Clear(Color.White);
 rnd := new Random();
 for i := 1 to n do
 begin
cr := rnd.next(256);
cg := rnd.next(256);
cb := rnd.next(256);
sb := new SolidBrush(Color.FromArgb
      (cr,cg,cb));
y1 := PictureBox1.Height-1;
y2 := y1-trunc(a[i] * m)-1;
gr.FillRectangle(sb, x, y2, h, y1);
x := x + 2 * h;
 end;
end;
```

Работающее приложение:



3. Изменить свойства Text у компонента Button2 на «Диаграмма со случайными данными».

4. Написать обработчик события Click для компонента Button1, в котором диаграмма строится с помощью прямоугольников.

4.1. Найти максимальный элемент в массиве *max*.

4.2. Рассчитать масштабный коэффициент:

$$m = \frac{PictureBox1.Height}{\max}.$$

4.3. В цикле строить n прямоугольников одинаковой ширины. Ширина прямоугольника

$$h = \frac{PictureBox1.Width}{2n+1}.$$

5. Обработчик для компонента Button2 будет отличаться от обработчика для компонента Button1 только способом получения элементов массива.

5.1. Массив должен быть описан в разделе var: `a: array[1..10] of integer;`

5.2. Элементы массива со значениями от 20 до 100 можно получать следующим образом:

```
rnd := new Random();
for i := 1 to n do
 a[i] := rnd.next(80) + 20;
```

4.4. Анимация

Эффект анимации достигается за счет того, что перед взглядом пользователя происходит быстрая смена изображений. Каждый из кадров анимации остается на экране очень небольшой промежуток времени.

Для замера интервалов времени можно использовать компонент *таймер*.

Он расположен на панели **Компоненты** и представлен в виде Timer, имя объекта Timer. Компонент Timer помещается в отдельную область ниже формы и получает имя TimerN, где N — номер 1, 2, 3... (пример 4.7).

Некоторые свойства компонента Timer приведены в таблице:

Свойство	Назначение
Enabled	Значение true обозначает, что таймер запущен
Interval	Время в миллисекундах, через которое происходит срабатывание таймера и вызов обработчика Tick

Компонент имеет единственный обработчик — Tick, в котором описываются действия, происходящие по истечении интервала срабатывания таймера.

Пример 4.8. Создать проект, в котором самолет будет пролетать над городом.

Этапы выполнения задания

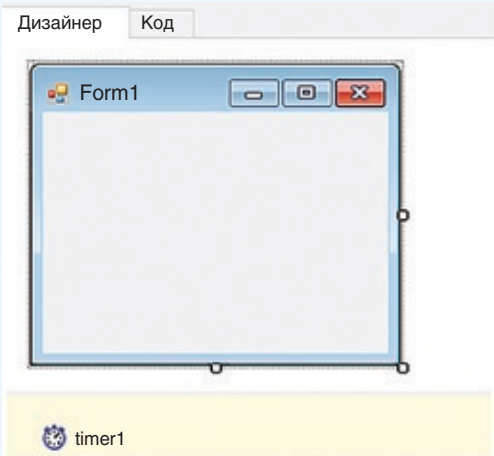
1. Поместить на форму компоненты PictureBox и Button, добавить компонент Timer.

2. Загрузить изображение города в компонент как фон формы (свойство формы BackgroundImage).

3. Установить прозрачный цвет фона для компонента PictureBox1 (значение Transparent у свойства BackColor). Установить режим изменения размера — AutoSize (свойство SizeMode).

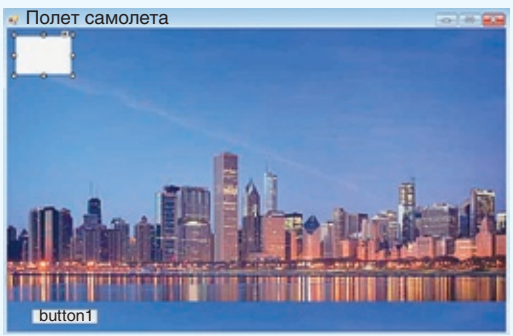
4. Написать обработчик события Load для формы и описать начальное

Пример 4.7. Компонент *таймер*:



Компонент Timer не виден при работе приложения, поэтому он размещается в области, специально предназначенной для невизуальных компонентов.

Пример 4.8. Форма на этапе конструирования:



Обработчик события Load для формы:

```
procedure Form1.Form1_Load (sender:
Object; e: EventArgs);
begin
 PictureBox1.Load ('plane.png');
 x := -PictureBox1.Width;
 y := 20;
 PictureBox1.Left := x;
 PictureBox1.Top := y;
end;
```

Пример 4.8. *Продолжение.*

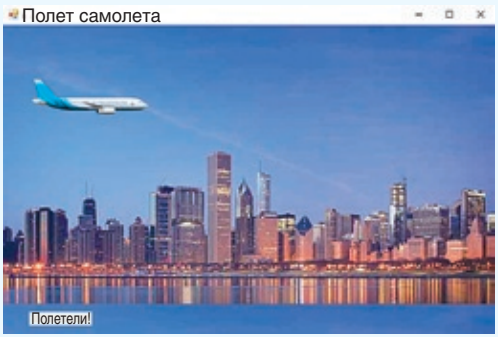
Обработчик события Click для компонента Button1:

```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
begin
 Timer1.Enabled := True;
end;
```

Обработчик события Tick для компонента Timer1:

```
procedure Form1.timer1_Tick
(sender: Object; e: EventArgs);
begin
 x := x + 1;
 PictureBox1.Left := x;
 if PictureBox1.Left > Width +
    + PictureBox1.Width then
  x := -PictureBox1.Width;
end;
```

Работающее приложение:



Если при запуске анимации возникает мерцание, то включить двойную буферизацию.

положение самолета, указав координаты верхнего левого угла PictureBox1 за пределами формы. Загрузить в PictureBox1 изображение из файла с рисунком самолета.

5. Изменить свойства Text у компонента Button1 на «Полетели!».

6. Установить значение False у свойства таймера Enabled в инспекторе объектов.

7. Установить в инспекторе объектов время срабатывания таймера, равным 10.

8. Написать обработчик события Click для компонента Button1, запустить таймер.

9. В инспекторе объектов установить прозрачность для компонента PictureBox1.

10. Написать обработчик события Tick и менять в нем значение свойства Left у компонента PictureBox1. Если самолет вылетел за границу, то вернуть его в начальное положение.

Двойная буферизация позволяет сделать анимацию более плавной, поскольку все операции рисования сначала выполняются в памяти, а лишь затем на экране компьютера. После завершения всех операций рисования содержимое буфера копируется из памяти непосредственно на связанную с ним область экрана.

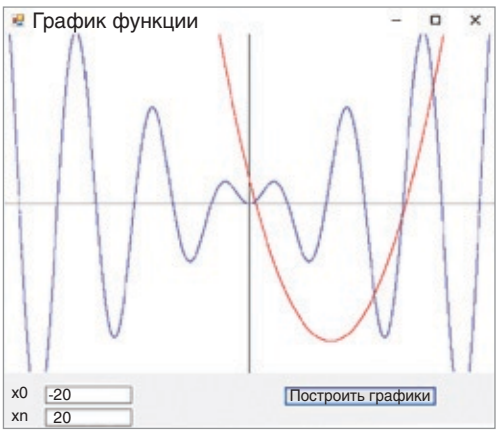
1. Какой компонент используется для размещения изображений?
2. Какое свойство позволяет загрузить готовое изображение в компонент PictureBox?
3. С помощью какого свойства можно установить прозрачный фон для изображения?
4. Какой класс представляет методы для рисования графических примитивов?
5. Какой метод канвы позволяет закрасить пиксель?
6. Какой компонент используется для отсчета времени?

Упражнения

1 Добавьте в проект из примера 4.4. еще одно животное (например, белку). Для размещения белки нужно добавить еще одну кнопку. Местоположение определить случайным образом в верхушке какого-либо дерева.

2 Добавьте в проект 4.5 перечисленные возможности.

1. Оси координат со стрелками и подписями.
2. Единичный отрезок на оси X.
3. График функции $y = 0{,}3x^2 - 4x + 2$ в той же системе координат красным цветом.



Постройте в одной системе координат графики следующих функций. Каждую кривую рисовать своим цветом. Для ввода значений параметров *a*, *b* и *c* добавьте на форму компоненты Text:

1. $y = ax^2$, $y = b\cos x^2$;
2. $y = ax^2 + bx + c$, $y = \frac{1}{x^2 + 3}$;
3. $y = ax^3$, $y = \frac{x}{(x+3)^2}$, $y = bx\ sin\ x$.

3 Измените проект из примера 4.6 так, чтобы строилась линейчатая диаграмма (столбики расположены горизонтально).

4* Измените проект из примера 4.6 так, чтобы значения в массив можно было вводить. Для этого числа необходимо записывать в компонент TextBox, считывать строку из чисел и пробелов, выделять цифры и преобразовывать их в числовые значения.

5 Добавьте в проект из примера 4.8 кнопку «Стоп», при нажатии на которую самолет остановится. *После остановки самолета должен появиться парашют и опуститься вниз.

6* Создайте анимацию движения Луны вокруг Земли. Для расчета координат верхнего левого угла PictureBox1, содержащего Луну, можно воспользоваться параметрическим уравнением окружности: $x = R\sin(t)$, $y = R\cos(t)$, где R — радиус, t — параметр, изменяющий свое значение от 0 до 2π.

§ 5. Создание приложений

Пример 5.1. Рекомендации по созданию оконных приложений.

1. В приложении рекомендуется разместить главное меню и инструментальную панель быстрых кнопок, дублирующих основные разделы меню.
2. Желательно, чтобы объекты приложения обладали контекстными меню, появляющимися при нажатии правой клавишей мыши на объекте.
3. Для объектов рекомендуется прописать подсказки, всплывающие при наведении указателя мыши на объект.
4. Рекомендуется реализовать строку состояния, используемую для выдачи различной информации.
5. При нажатии клавиши F1 должен загружаться файл справки.
6. В программе желательно реализовать возможность настройки и сохранения настроек, чтобы при следующем сеансе работы их не пришлось устанавливать заново.
7. Если результат работы приложения зависит от каких-либо параметров, обязательно укажите значения по умолчанию. Они позволят ускорить взаимодействие пользователя с программой, а также являются примером того, в каком формате данные следует вводить.

Мощным воздействием на психику человека является цвет, поэтому с ним нужно обращаться очень осторожно. Нужно стремиться использовать ограниченный набор цветов и уделять внимание их правильному сочетанию. Восприятие цвета у человека очень индивидуально, поэтому не стоит навязывать всем свое видение цвета. Желательно, чтобы основной цвет формы был нейтральным (например, у большинства приложений Microsoft это светло-серый цвет).

5.1. Разработка оконных приложений

Создание любого оконного приложения осуществляется, как правило, в три этапа:

1. Создание интерфейса приложения, т. е. средств взаимодействия пользователя с программой.

2. Разработка сценария работы будущего приложения. На этом этапе определяют, какая информация будет выводиться на экран, какие события будут происходить при использовании различных компонентов, как приложение должно завершить работу, какие результаты и в каком виде сохранить и т. д.

3. Разработка алгоритма решения поставленной задачи.

Большинство приложений в операционной системе Windows выглядят и ведут себя сходным образом. Компания Microsoft предложила рекомендации для разработки программного обеспечения[1], направленные на то, чтобы пользователь не тратил время на освоение нюансов пользовательского интерфейса новой программы, а сразу начал продуктивно ее использовать. Эти рекомендации основаны на психофизиологических особенностях человека и существенно облегчат жизнь будущим пользователям вашей программы.

Приведем некоторые рекомендации по разработке графического интерфейса оконных приложений (пример 5.1).

[1] https://docs.microsoft.com/ru-ru/windows/uwp/design/basics/design-and-ui-intro

5.2. Стандартные диалоги

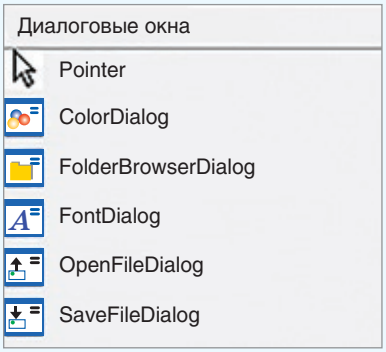
Практически любое приложение Windows использует стандартные диалоги, встроенные в операционную систему, для открытия и сохранения файлов, выбора атрибутов шрифта или установки цвета, поиска текста, печати. В библиотеку VCL включены компоненты, реализующие соответствующие окна Windows. Они размещены на панели **Диалоговые окна** (пример 5.2). В примере 5.3 приведен перечень компонентов для реализации стандартных диалогов.

Объекты на странице **Диалоговые окна** невидимы во время выполнения, поэтому они размещаются в специальной области под формой (пример 5.4). Внешний вид окна диалога зависит от версии Windows.

Вызов и обработка диалогов происходит программно. Для всех диалогов определен метод `ShowDialog()` (пример 5.5). С помощью этого метода происходит открытие окна соответствующего диалога. В свойствах компонента-диалога запоминается выбор пользователя, который затем можно обработать.

Диалоги для открытия и сохранения файлов используются в различных приложениях. Основное свойство компонентов OpenDialog и SaveDialog, в котором возвращается в виде строки имя файла, — это свойство **FileName**. Если задать данное свойство на этапе конструирования в окне инспектора объектов, то при открытии диалога

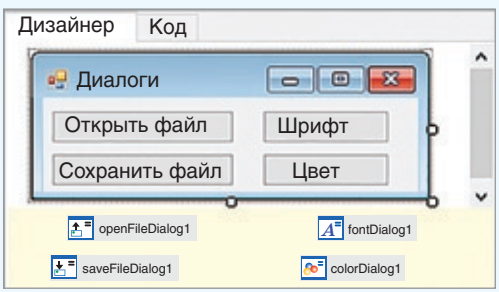
Пример 5.2. Панель **Диалоговые окна**.



Пример 5.3. Список некоторых стандартных диалогов.

Компонент	Назначение
OpenFileDialog	Создание окна диалога «Открыть файл»
SaveFileDialog	Создание окна диалога «Сохранить файл»
FontDialog	Создание окна диалога «Шрифт» — выбор атрибутов шрифта
ColorDialog	Создание окна диалога «Цвет» — выбор цвета

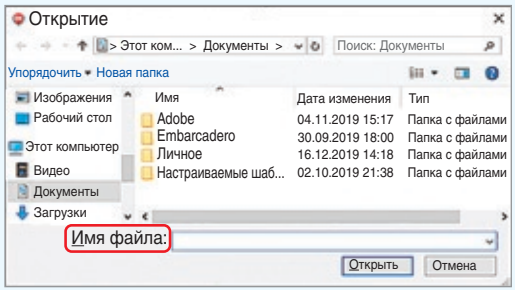
Пример 5.4. Диалоговые компоненты и кнопки для их вызова на форме:



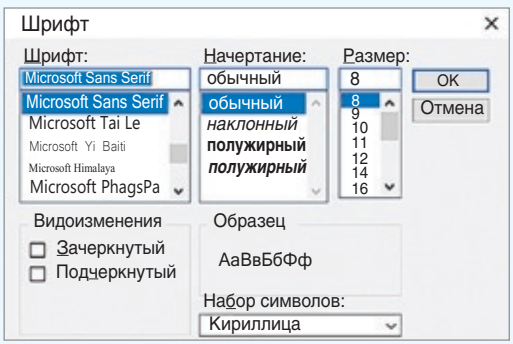
Пример 5.5. Стандартное обращение к диалогу:

```
<имя диалога>. ShowDialog();
```

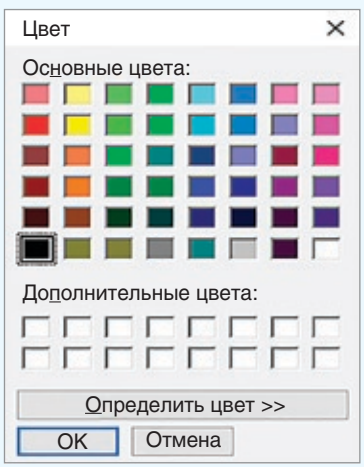
Пример 5.6. Стандартный диалог для открытия файла:



Пример 5.7. Стандартный диалог для настроек шрифта:



Пример 5.8. Стандартный диалог для выбора цвета:



оно будет появляться в строке Имя файла (пример 5.6).

Для вызова стандартного окна установки атрибутов шрифта можно использовать компонент FontDialog (пример 5.7). В окне **Шрифт** пользователь может выбрать имя шрифта, его стиль, размер. Основное свойство компонента — Font.

Для вызова стандартного окна установки цвета используется компонент ColorDialog (пример 5.8). В нем можно выбрать цвет из базовой палитры. Основное свойство компонента ColorDialog — **Color**. Это свойство соответствует тому цвету, который пользователь выбрал в диалоге.

5.3. Создание меню

Практически любое приложение должно иметь меню, которое дает удобный доступ к функциям программы. Существует несколько типов меню:

- **главное меню** с выпадающими списками разделов;
- **каскадные меню**, в которых разделу первичного меню ставится в соответствие список подразделов;
- **контекстные меню**, появляющиеся при нажатии правой клавишей мыши на объекте.

В PascalABC.Net меню создаются компонентами MenuStrip (главное меню) и ContextMenuStrip (контекстное меню), расположенными на панели **Меню и панели инструментов**. Во время выполнения программы сами компоненты не видны, поэтому раз-

мещаются в специальной области под формой (пример 5.9). На этапе выполнения программы главное меню будет помещено на свое стандартное место — наверху формы, контекстное меню появится только после нажатия правой кнопки мыши по тому компоненту, к которому оно относится.

Для добавления новых пунктов меню нужно кликнуть левой клавишей мыши в верхней части формы (там, где обычно располагается меню). Затем заполнить ячейки, соответствующие пунктам меню (пример 5.10).

Каждый пункт меню является отдельным объектом. Список всех компонентов, относящихся к меню, можно увидеть в выпадающем списке в инспекторе объектов. Названия пунктов меню прописываются в свойстве Text в окне инспектора объектов (пример 5.11).

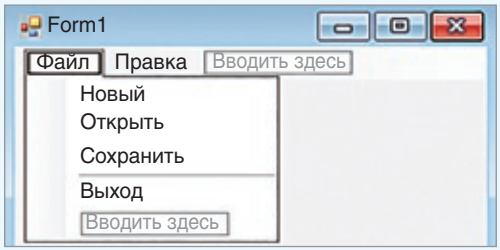
Для каждого пункта меню основным событием является событие Click.

Создание контекстных меню аналогично созданию главного меню. Сначала нужно выбрать компонент на нижней панели, а затем заполнить ячейки. Для того чтобы при щелчке правой кнопкой мыши на некотором компоненте появлялось контекстное меню, нужно написать имя контекстного меню в свойстве ContextMenuStrip для выбранного компонента (пример 5.12).

Написание обработчиков для меню и диалогов будет рассмотрено в следующих пунктах.

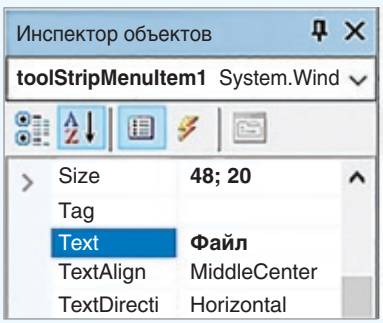
Пример 5.9. Меню:

Пример 5.10. Редактирование меню:

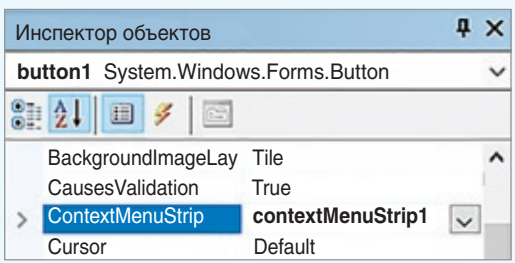


Если в качестве значения свойства Caption ввести «—», то вместо пункта меню появится разделитель.

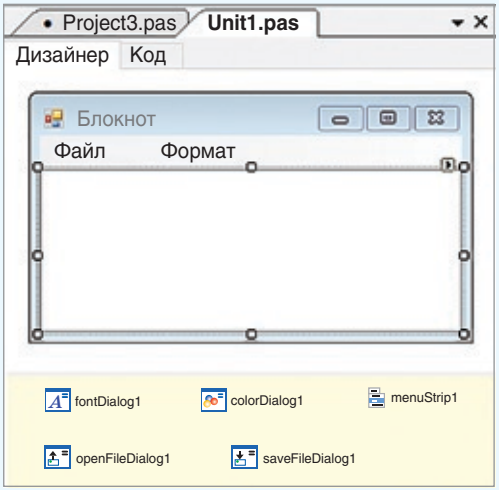
Пример 5.11. Название пункта меню в инспекторе объектов:



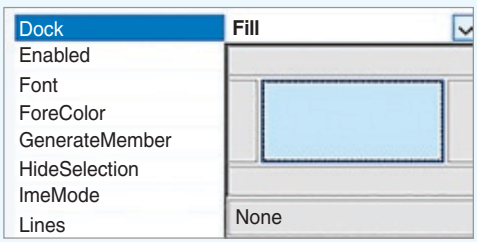
Пример 5.12. Контекстное меню для компонента Button1:



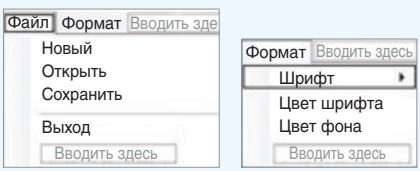
Пример 5.13. Форма на этапе конструирования:



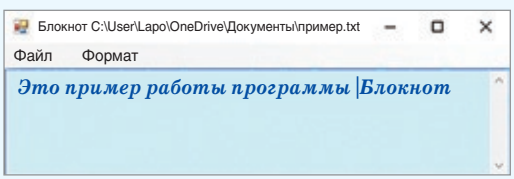
Пример 5.14. Настройка клиентской области:



Пример 5.15. Структура меню:



Пример 5.16. Работающее приложение:



5.4. Создание приложения «Блокнот»

Программа Блокнот должна давать возможность открыть и сохранить текстовый файл, выбрать цвет текста и цвет фона.

Разместить на форме (пример 5.13) следующие компоненты:

- рабочая область для текста — TextBox1;
- диалоги работы с файлами — OpenFileDialog1, SaveFileDialog1;
- диалоги для настройки внешнего вида приложения — FontDialog1, ColorDialog1;
- главное меню — MenuStrip1.

Компонент TextBox1 предназначен для набора и редактирования текста. Текст может набираться в несколько строк, поэтому нужно установить значение true для свойства MiltiLine. Для того чтобы компонент занимал всю клиентскую часть формы, необходимо установить у свойства Dock значение Fill (пример 5.14). Установить значение Vertical для свойства ScrollBars (вертикальная полоса прокрутки).

Структура меню представлена в примере 5.15. Для написания обработчиков пунктов меню нужно в инспекторе объектов выбрать соответствующий пункт меню, перейти на вкладку Events и выбрать событие Click. Поскольку событие Click является событием по умолчанию, то двойной клик по пункту в редакторе меню создаст процедуру-обработчик.

Окно работающего приложения показано в примере 5.16.

Обработчики событий для каждого из пунктов меню представлены в примере 5.17.

Для сохранения и загрузки файлов опишем глобальную переменную F_N:

```
var F_N: String;
```

Обработчик пункта меню **Новый** (StripMenuItem4) очищает строки компонента TextBox1 от введенного ранее текста.

Обработчики пунктов меню **Открыть** (StripMenuItem5) и **Сохранить** (StripMenuItem6) работают с файлом. Имя файла добавляется к заголовку окна.

Обработчик пункта меню **Выход** (StripMenuItem8) закрывает главную форму проекта.

Обработчик пункта меню **Шрифт** (StripMenuItem9) приписывает шрифту, связанному с компонентом TextBox1, свойства, выбранные пользователем.

Обработчики пунктов меню **Цвет текста** (StripMenuItem10) и **Цвет фона** (StripMenuItem11) устанавливают для TextBox1 цвета текста и фона, выбранные пользователем.

Для компонента TextBox определены следующие действия: Копировать (Ctrl + C), Вырезать (Ctrl + X), Вставить (Ctrl + V), Отменить (Ctrl + Z).

5.5. Создание приложения «Графический редактор»

Программа «Графический редактор» должна давать возможность открыть и сохранить файл, выбрать цвет линии и цвет фона, установить толщину линии. Рисование производится выбранным цветом линии при нажатой левой клавише мыши. Клик правой клавишей мыши внутри замкнутой области используется для заливки ограниченной области выбранным цветом фона.

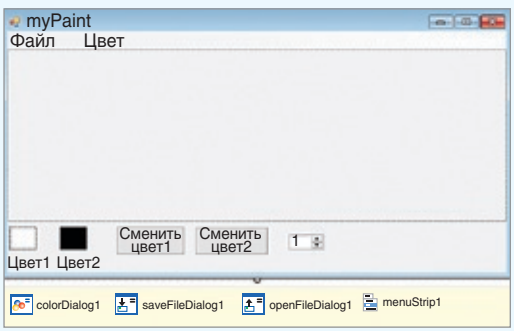
Пример 5.17. Обработчики событий для пунктов меню:

```
var s:string;

procedure
Form1.toolStripMenuItem4_Click
(sender: Object; e: EventArgs);
begin
 //Файл — Новый
 TextBox1.Clear;
end;

procedure
Form1.toolStripMenuItem5_Click
(sender: Object; e: EventArgs);
begin
//Файл — Открыть
 openFileDialog1.ShowDialog();
 s := openFileDialog1.FileName;
 Text := 'Блокнот ' + s;
 TextBox1.Lines := ReadAllLines(s);
end;

procedure
Form1.toolStripMenuItem6_Click
(sender: Object; e: EventArgs);
begin
//Файл — Сохранить
 saveFileDialog1.ShowDialog();
 F_N := saveFileDialog1.FileName;
 WriteAllLines(F_N,
   TextBox1.Lines);
 Text := 'Блокнот ' + F_N;
end;

procedure
Form1.toolStripMenuItem8_Click
(sender: Object; e: EventArgs);
begin
 //Файл — Выход
 close;
end;
```

Пример 5.17. *Продолжение.*

```
procedure Form1.
toolStripMenuItem9_Click
(sender: Object; e: EventArgs);
begin
 //Формат — Шрифт
 fontDialog1.ShowDialog();
 TextBox1.Font := fontDialog1.
Font;
end;
procedure Form1.
toolStripMenuItem10_
Click(sender: Object; e:
EventArgs);
begin
 //Формат — Цвет текста
 colorDialog1.ShowDialog();
 TextBox1.ForeColor :=
   colorDialog1.Color;
end;
procedure Form1.
toolStripMenuItem11_
Click(sender: Object; e:
EventArgs);
begin
 //Формат — Цвет фона
 colorDialog1.ShowDialog();
 TextBox1.BackColor :=
   colorDialog1.Color;
end;
```

Пример 5.18. Форма на этапе конструирования:



Сначала спроектируем форму, разместив на ней следующие компоненты (пример 5.18):

- область для рисования — PictureBox;
- компоненты, отображающие выбранный цвет для рисования и цвет фона — Panel1, Panel2;
- кнопки для смены цвета;
- компонент выбора цвета — ColorDialog1;
- компонент для выбора толщины линии — numericUpDown1 (панель компонентов **Стандартные элементы управления**);
- главное меню — menuStrip1 и компоненты для работы с файлами — OpenFileDialog1, SaveFileDialog1.

На этапе конструирования установить значение свойства BackColor у компонентов Panel1 и Panel2 — Black и White соответственно.

У свойств Value и Minimum для компонента numericUpDown1 установить значение 1.

Структура меню показана в примере 5.19. Создание рисунка и загрузка файла в приложение приведены в примере 5.20.

В обработчике события Load для формы прописаны первоначальные установки для создания графического объекта, заданы параметры кисти и карандаша.

В обработчике события MouseDown для компонента PaintBox1 задаем переменной m_d значение true — кнопка нажата. Здесь же запоминаем координаты точки, поскольку от этой точки начнем строить линию. В обработчи-

ке MouseUp — значение переменной m_d = false — кнопка не нажата.

Для отслеживания траектории движения мыши по компоненту PaintBox1 создаем обработчик события MouseMove. Если кнопка нажата, то можем строить линию. Параметры e.x, e.y возвращают координаты точки, в которой произошло нажатие кнопки.

Для перемещения мыши нужно использовать метод DrawLine (x1, y1, e.x, e.y) — рисование линии, соединяющей две точки. После прорисовки обновляем координаты.

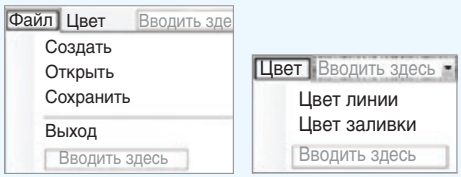
Толщина линии определяется значением свойства Value для компонента numericUpDown1. Обработчик события — ValueChanged.

Обработчики событий для компонентов OpenFileDialog1, SaveFileDialog1 вызываются из соответствующих пунктов меню и аналогичны обработчикам, описанным для программы Блокнот. Для сохранения и загрузки файлов нужно описать глобальную строковую переменную FileName. Приложение может сохранять и загружать файлы формата BMP.

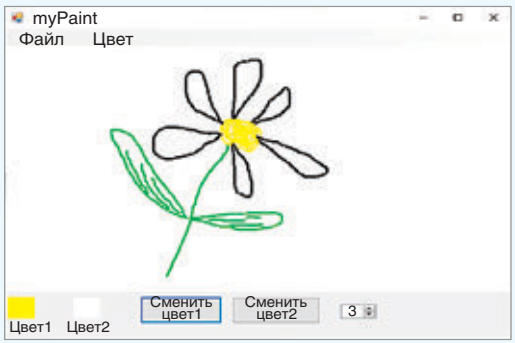
С помощью компонента ColorDialog1 можно выбрать цвет линии или заливки. Пункты меню **Цвет** позволяют выбрать цвет линии или заливки соответственно.

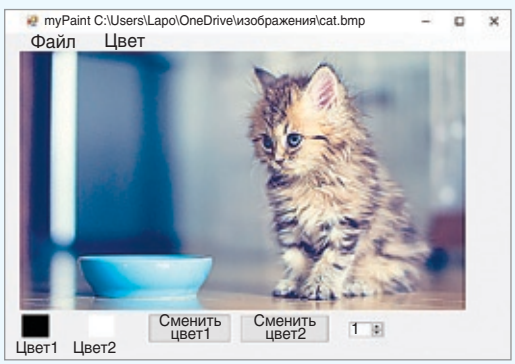
В примере 5.21 приведено описание глобальных переменных, которые используются для создания приложения «Графический редактор». Обработчики всех описанных событий приведены в приложении.

Пример 5.19. Структура меню:



Пример 5.20. Работающее приложение:

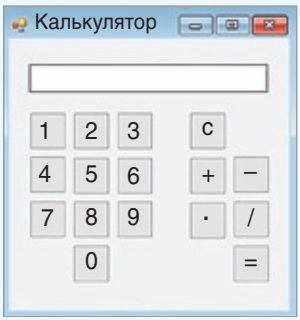




Пример 5.21. Описание глобальных переменных.

```
var gr: Graphics;
   bm: Bitmap;
   p_c: Pen;
   s_b: SolidBrush;
   c_f, c_b: Color;
   w: decimal;
   x1, y1, x2, y2: integer;
   m_d : boolean;
   F_N :string;
```

Пример 5.22. Форма на этапе конструирования:



Пример 5.23. Обработчики событий:

```
//описание глобальных переменных
var n1, n2: integer;
  znak: char;
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
begin
 //приписывание цифры к числу
 TextBox1.Text :=TextBox1.Text+'1';
end;
procedure Form1.button9_Click
(sender: Object; e: EventArgs);
begin
 //приписывание цифры к числу
 TextBox1.Text :=TextBox.Text1+'2';
end;
```

Для остальных цифровых кнопок нужно изменить только '1' на соответствующую цифру.

```
procedure Form1.button12_Click
 (sender: Object; e: EventArgs);
begin
 n1 := StrToInt(TextBox1.Text);
 //запоминание знака операции
 znak := '+';
 TextBox1.Clear;
end;
```

Для остальных кнопок со знаками арифметических действий нужно изменить только '+' на соответствующий знак.

5.6. Создание приложения «Калькулятор»

Создание калькулятора начнем с конструирования формы. На ней нужно разместить: поле TextBox для ввода/вывода чисел, 10 кнопок с цифрами, 4 кнопки с арифметическими действиями, кнопку «=» и кнопку «C» — очистить (пример 5.22).

При нажатии на кнопку с цифрой программа должна дописать эту цифру к числу в поле TextBox. При нажатии на кнопку с арифметическим действием нужно запомнить число, которое в данный момент находится в поле TextBox, и очистить поле для ввода второго числа. Числа будем хранить в двух переменных *n*1, *n*2 типа integer. Знак операции будем хранить в переменной *znak* типа **char**. Переменные описываются как глобальные. При нажатии на кнопку «=» выполняется арифметическое действие и выводится результат.

Кнопки могут содержать рисунок на поверхности (например, изображения с цифрами). Свойство для размещения рисунка — BackgroundImage.

Установить значение FixedSingle для свойства FormBorderStyle формы. В этом случае граница формы не позволит менять ее размеры.

Коды процедур-обработчиков приведены в примере 5.23.

Для каждой кнопки на форме нужно создать обработчик события Click.

Обработчики событий для всех цифровых кнопок будут идентичны.

Обработчики для кнопок арифметических действий будут отличаться

только значением запоминаемой операции.

Основные вычисления происходят в обработчике кнопки «=». Преобразуем в число n2 значение поля Edit и выполняем арифметическую операцию в зависимости от значения переменной znak. После этого обнуляем переменные.

В обработчике кнопки «C» (от англ. clear — очистить) происходит обнуление переменных и очистка поля Edit.

Созданный калькулятор имеет большое количество ограничений в своей работе, поскольку рассчитан на вычисления только с натуральными числами.

Пример 5.23. *Продолжение.*

```
procedure Form1.button16_Click
(sender: Object; e: EventArgs);
begin
 n2 := StrToInt(TextBox1.Text);
 case znak of
  '+': TextBox1.Text:= IntToStr(n1 + n2);
  '-': TextBox1.Text:= IntToStr(n1 - n2);
  '*': TextBox1.Text:= IntToStr(n1 * n2);
  '/': TextBox1.Text:=IntToStr(n1 div n2);
 end;
 n1 := 0;  n2 := 0;
 znak := ' ';
end;
```

```
procedure Form1.button15_Click
(sender: Object; e: EventArgs);
begin
 TextBox1.Clear;
 n1 := 0;  n2 := 0;
 znak :=' ';
end;
```

Упражнения

1 Дополните проект Блокнот следующими возможностями:

1. Пункт меню Сохранить файл заменить двумя: Сохранить и Сохранить как...
2. Добавить Пункт меню Переносить по словам.
3. Добавьте возможность управления полосами прокрутки: установить горизонтальную, вертикальную, обе.

4*. Добавьте следующие диалоги: Параметры страницы, Печать, Поиск, Замена и соответствующие пункты в меню.

5. Добавьте контекстное меню для управления компонентом Memo. Команды контекстного меню, дублирующие команды основного меню, должны вызывать те же обработчики.
6. Предложите свои возможности.

2 Для проекта Графический редактор добавьте следующие возможности:

1. Рисовать отрезки, овалы и прямоугольники.
2. Предложите свои возможности.

3 Для проекта Калькулятор добавьте следующие возможности:

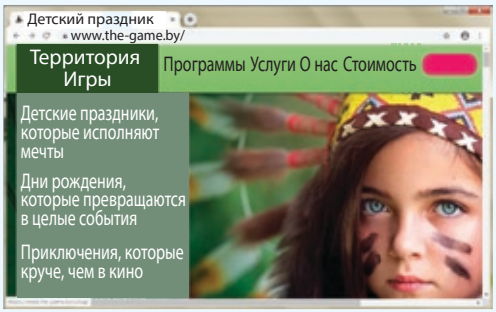
1. Возможность работы с вещественными числами.
2. Возможность вычислять значения функций: извлечение квадратного корня, тригонометрические функции (для градусов и радиан) и др.

3*. Возможность перевода чисел в другие системы счисления.

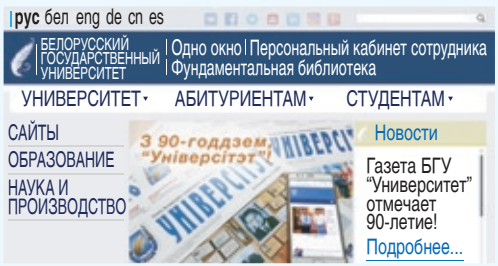
Глава 2
ОСНОВЫ ВЕБ-КОНСТРУИРОВАНИЯ

§ 6. Веб-конструирование. Основные понятия

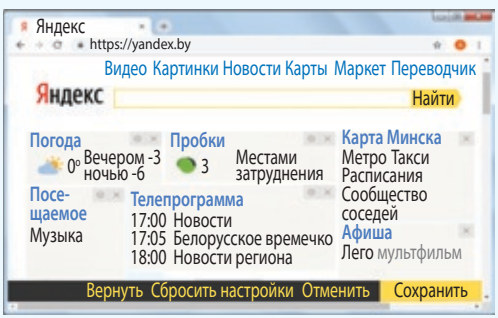
Пример 6.1. Сайт компании, организующей детские праздники.



Пример 6.2. Сайт Белорусского государственного университета.



Пример 6.3. Поисковая система Яндекс.



6.1. Веб-сайт

Веб-сайт представляет собой группу веб-страниц, связанных между собой гиперссылками и объединенных общей темой и оформлением.

В зависимости от содержания веб-сайты могут быть:

1. Презентационные (сайты-визитки) — реклама определенных акций, мероприятий, событий, продвижение конкретных технологий, сервисов или услуг (пример 6.1).

2. Корпоративные — представляют компанию, предприятие, организацию. Задача таких сайтов — создание положительного имиджа, а также продвижение товаров или услуг (пример 6.2).

3. Онлайн-сервисы — направлены на удовлетворение повседневных и бытовых нужд обычного пользователя: поиск информации, работа с электронной почтой и др. (пример 6.3).

4. Электронные магазины — ориентированы на получение прибыли от продажи товаров (пример 6.4).

Типы веб-сайтов в зависимости от технологии создания:

1. Статические — хранятся на сервере и отображаются в браузере в одном и том же виде (пример 6.5).

2. Динамические — полностью или частично генерируются на сервере или в браузере в процессе исполнения запроса пользователя (пример 6.6).

По взаимодействию пользователя с ресурсами веб-страниц веб-сайты можно разделить на:

1. Пассивные — информацию на таких сайтах можно только просматривать (пример 6.5).

2. Интерактивные — пользователь имеет возможность обмениваться данными с сервером, участвовать в интерактивном диалоге (пример 6.6).

6.2. Язык гипертекстовой разметки документа HTML. Структура HTML-документа. Теги и атрибуты. Гиперссылки

Веб-страница представляет собой гипертекстовый документ, созданный на языке HTML (HyperText Markup Language — язык разметки гипертекста).

Язык HTML описывает структуру html-документа, позволяет задать базовые параметры веб-страницы, свойства и местоположение объектов веб-страницы.

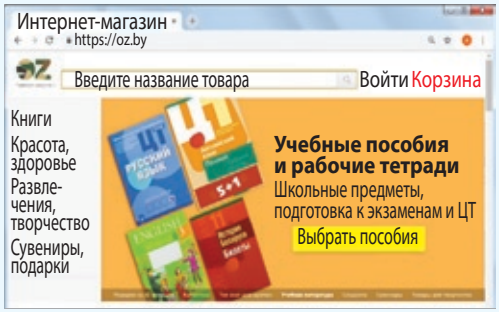
Html-документ — файл с расширением **.html**. Такой файл может быть открыт в браузере как веб-страница.

Основные компоненты языка HTML — теги, атрибуты и их значения.

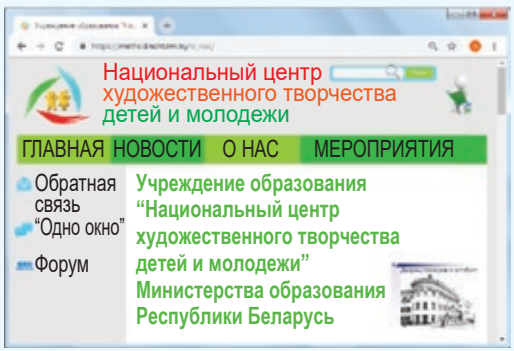
Теги — набор специальных символов языка HTML, которые идентифицируют html-документ, определяют разделы веб-страницы и положение элементов на веб-странице.

Обычно тег — это контейнер из пары тегов (открывающего и закрывающего). Записываются теги в треугольных скобках: <>. Закрывающий тег

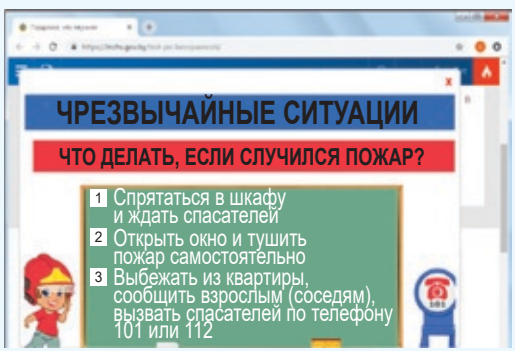
Пример 6.4. Сайт интернет-магазина OZ.



Пример 6.5. Сайт Национального центра художественного творчества детей и молодежи.



Пример 6.6. Интерактивная страница на сайте МЧС РБ с тестом по безопасности «Гордимся, что научили».



Язык HTML постоянно развивается. С 2014 г. рекомендуется использовать версию HTML5. Эта версия содержит теги canvas, audio и video, позволяющие добавлять графические, звуковые и видеоматериалы.

Пример 6.7. Структура HTML-документа.

```
<html>
 <head> <  ! -- Заголовок -->
  <title>  Название страницы
  </title>
  <meta charset = «utf-8»>
   <! -- установка кодовой таблицы utf-8-->
 </head>
 <body> <! -- тело -->
  <h1>Заголовок статьи</h1>
  <p>Абзац статьи</p>
 </body>
</html>
```

Пример 6.8. Написание и отображение в браузере тега <title>.

```
<head>
 <title>Пример  веб-страницы
 </title>
</head>
```

Для повышения качества html-кода и обеспечения совместной работы над веб-сайтами принято руководствоваться определенными правилами оформления и форматирования:

1. Использовать нижний регистр, несмотря на то что язык html нечувствителен к регистру.
2. Записывать код так, чтобы строки полностью помещались на экране.
3. Выделять новую строку для каждого открывающего тега и ставить отступы для каждого дочернего (вложенного) элемента.
4. При создании веб-страниц учитывать разницу в отображении html-кода в различных браузерах.

отличается от открывающего только наличием символа /.

Код веб-страницы имеет определенную структуру:

1. Контейнер <html> </html> идентифицирует код как html-документ. В него вкладываются контейнеры <head> </head> и <body> </body>.
2. Контейнер <head> содержит не отображаемую на веб-странице информацию — название html-документа, теги для поисковых машин и др.
3. Контейнер <body> определяет содержание веб-страницы (контент), отображаемое браузером.

(Рассмотрите пример 6.7.)

Часто при написании html-кода возникает необходимость вставки поясняющих комментариев. Начинаются они с <!-- и заканчиваются -->. Все, что находится внутри этого тега, отображаться на веб-странице не будет.

В заголовоке html-документа размещаются:

1. Обязательный контейнер <title> </title>, который определяет название веб-страницы и отображается на вкладке браузера (пример 6.8).
2. Тег <meta>, устанавливающий кодировку документа.

Атрибуты расширяют возможности тегов, задают свойства объектов на веб-странице. Например, можно выровнять абзац или задать размер изображения.

Значение атрибута определяет конкретные свойства объектов на веб-странице. Например, если для тега используется атрибут выравнивания, то он может иметь значение left или right.

Атрибуты и их значения записываются внутри открывающего тега (пример 6.9).

Основным свойством гипертекста является способность соединять в единое целое информацию, представленную в цифровой форме (текст, аудио- и видеофайлы, графику, анимацию, html-документы). Главным при этом становится указание на то, что с чем и каким образом должно быть связано. На веб-страницах такие связи реализуются с помощью **гиперссылок**. Гиперссылка состоит из двух частей: указателя ссылки и адресной части ссылки. Указатель ссылки — это то, что мы видим на веб-странице (текст или рисунок). Текстовый указатель обычно выделяется синим цветом и подчеркиванием.

Пример 6.9. Синтаксис языка HTML.

```
<body bgcolor = "gray" text = "white">
  Содержимое
</body>
```

— пара тегов;

— атрибуты;

— значения атрибутов;

— содержимое тега-контейнера.

При оформлении гиперссылок рекомендуется следовать правилам:

1. Ссылка должна быть легко узнаваема.
2. Ссылка не должна нарушать целостность текста. Лишние ссылки мешают чтению.
3. По текстовому указателю ссылки должно быть понятно, куда она ведет.

Текст под ссылкой не должен быть длинным.

1. По каким принципам можно разделить сайты на группы?
2. На какие группы можно разделить сайты в зависимости от их содержания?
3. Какой язык используется при создании веб-страниц?
4. Какую структуру имеет html-документ?
5. Что такое тег, атрибут?
6. Для чего служат гиперссылки?

Упражнения

1 Откройте в браузере указанные сайты.

1. edu.gov.by
2. camelotmebel.by
3. careers.epam.by
4. calc.by

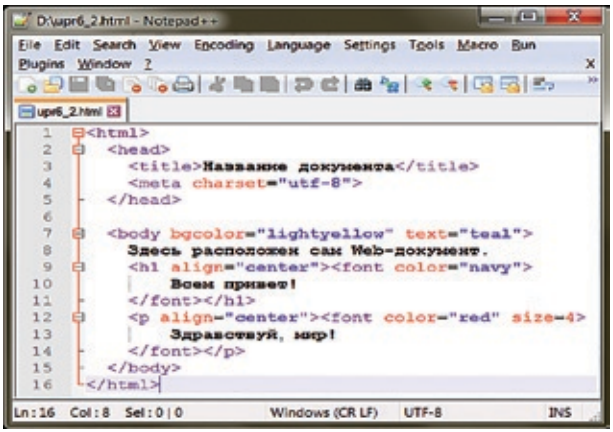
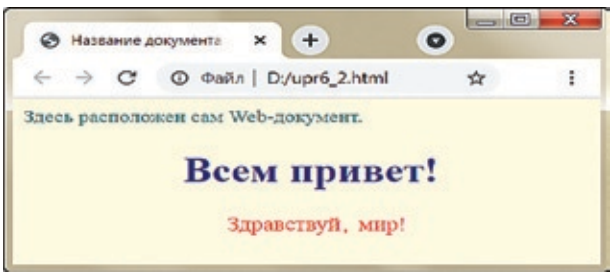
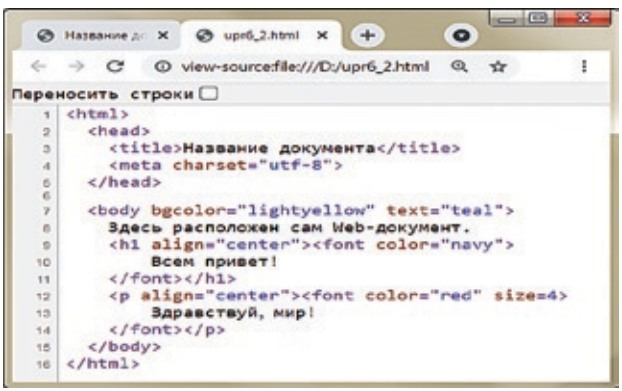
К какому типу можно отнести каждый из сайтов в зависимости от их содержания?

2 Откройте страницы веб-сайтов, воспользовавшись ссылками:

1. http://testy.by/quiz
2. https://stellarium-web.org

К какому типу можно отнести данные сайты по организации взаимодействия пользователя с ресурсами веб-страниц этих веб-сайтов?

3 Выполните указанные действия. Каким цветом при просмотре кода страницы в браузере выделяются теги, атрибуты, значения атрибутов, содержимое тегов?

Действие	Результат
Откройте файл в редакторе кода (например, Notepad++)	
Сохраните этот файл как html-документ и откройте его в браузере	
В контекстном меню открытой в браузере веб-страницы выберите **Просмотр кода страницы** или воспользуйтесь комбинацией клавиш Ctrl + U	

4 Выпишите в тетрадь теги, определяющие структуру html-документа из упражнения 3.

§ 7. Создание веб-страниц

7.1. Инструменты создания веб-страниц

Для создания веб-страниц могут использоваться различные средства, от самых простых до очень сложных. Выбор инструментов зависит от того, для кого и с какой целью создается сайт.

В качестве инструментов для создания веб-страниц можно использовать:

1. Текстовые редакторы (например, **Блокнот**). Создание веб-страниц в этом случае представляет собой трудоемкий процесс.

2. Редакторы с подсветкой html-кода. Такие редакторы обычно имеют интеллектуальный режим ввода (подсказка при вводе), что оптимизирует и упрощает работу (пример 7.1).

3. Визуальные веб-редакторы. С помощью таких редакторов можно создавать веб-страницу, не зная языка HTML. Содержание в процессе редактирования выглядит максимально похожим на конечную продукцию (пример 7.2).

4. Конструкторы сайтов — интернет-сервисы, которые предоставляют возможность быстро создать сайт с современным дизайном без знаний языка HTML и без использования специального программного обеспечения (пример 7.3).

После написания кода html-документа в каком-либо редакторе его требуется сохранить как файл с расширением **.html**.

Пример 7.1. Редакторы html-кода.

Notepad++	Бесплатный редактор с открытым кодом для Windows
Atom	Бесплатный кроссплатформенный (Windows, MacOS, Linux) редактор
Sublime Text	Кроссплатформенный редактор. Имеет бесплатную и платную версию

Пример 7.2. Визуальные веб-редакторы.

Web Page Maker	Бесплатная пробная версия. Предлагаются разнообразные шаблоны элементов навигации
Web Builder	Бесплатная пробная версия. Поддержка многих форматов видео
Adobe Dreamviewer	Бесплатная пробная версия. Сложно использовать без знания HTML

Пример 7.3. Конструкторы сайтов.

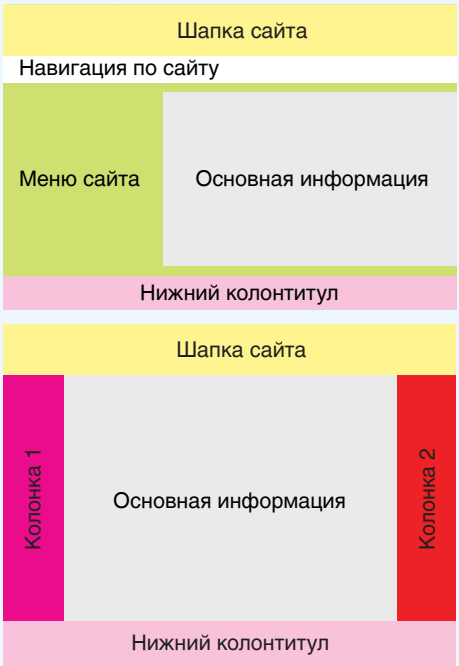
WIX

Бесплатные и платные тарифы. Конструктор постоянно обновляется и предлагает новые возможности.

ukit GROUP

Есть бесплатный пробный период. Много стильных шаблонов.

Пример 7.4. Элементы веб-страницы. Варианты расположения:



Версткой веб-страниц называется создание такого html-кода, который позволяет размещать элементы веб-страницы в нужных местах документа и отображать их в окне браузера согласно разработанному макету. Макет разрабатывается веб-дизайнером в графическом редакторе.

Нередко при создании веб-страницы возникает необходимость в использовании символов, которых нет в стандартной компьютерной клавиатуре либо которые не поддерживает кодировка html-документа. Такие символы называются спецсимволами HTML.

Чтобы разместить спецсимволы на веб-странице, необходимо указать их html-коды (см. *Приложение к главе 2*, с. 107).

7.2. Элементы оформления веб-страниц

Любая веб-страница содержит определенный набор стандартных элементов. Безусловно, этот набор может варьироваться в зависимости от тематической направленности, а также от целей и задач сайта. Создание таких элементов и проектирование их взаимного расположения является одной из главных задач веб-разработки.

Основные элементы оформления веб-страниц:

1. Заголовок (шапка сайта, header). Располагается в верхней части каждой страницы сайта. Может быть выполнен как в текстовом, так и в графическом варианте (часто это **логотип**).

2. Основная часть. Занимает бóльшую часть веб-страницы. Здесь размещается содержательный информационный текст и иллюстрации — контент (от. англ. *content* — содержание).

3. Элементы навигации (меню). Основное меню обычно располагается под шапкой, а вспомогательное — в левой части страницы. Представляет собой совокупность гиперссылок.

4. Нижний колонтитул (подвал, footer). Располагается на каждой странице сайта. Обычно здесь размещается информация о разработчике, контактная информация, иконки социальных сетей и т. д.

5. Боковые панели. Вертикальные полосы справа и/или слева от основной части. В них располагаются, например, ссылки, рекламные блоки, форма голосования и т. д.

(Рассмотрите пример 7.4.)

7.3. Текст на веб-странице

Как правило, основу контента веб-страниц составляет текстовая информация. Поэтому очень важно научиться вводить и форматировать блоки текста на странице. Для работы с текстом в языке HTML имеется большое количество тегов.

Обычно текст разделяют на абзацы. Это облегчает чтение большого текста. В языке HTML для создания абзаца используется контейнер <p>. При просмотре html-страницы в браузере абзацы отделяются небольшими интервалами (пример 7.5).

Для выравнивания текста в абзаце тег <p> поддерживает атрибут `align` (пример 7.6). Он может принимать одно из четырех значений:

- `left` — выравнивание текста по левому краю (по умолчанию);
- `center` — выравнивание текста по центру;
- `right` — выравнивание текста по правому краю;
- `justify` — выравнивание по ширине.

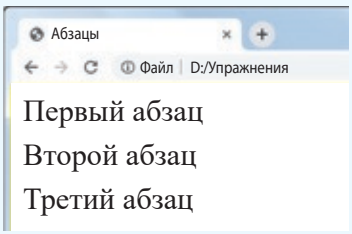
Для выделения разделов текста на веб-странице применяются заголовки. В языке HTML существует шесть уровней заголовков. Самым верхним уровнем является уровень 1 — тег <h1>, а самым нижним — уровень 6, тег <h6>. Содержимое заголовка первого уровня отображается самым крупным шрифтом жирного начертания, а заголовка последнего, шестого уровня —

Пример 7.5. Использование тега <p>.

Html-код:

```
<p>Первый абзац.</p>
<p>Второй абзац.</p>
<p>Третий абзац.</p>
```

Отображение в браузере:

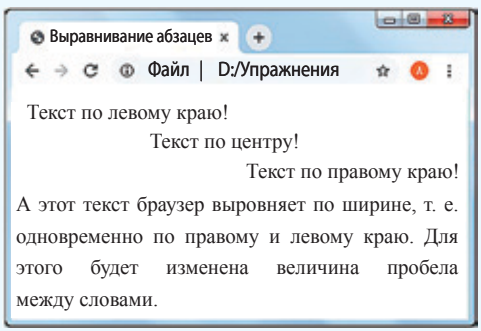


Пример 7.6. Выравнивание абзацев в html-документе.

Html-код:

```
<p align = "left">Текст по левому краю!
</p>
<p align = "center">Текст по центру!</p>
<p  align  =  "right">Текст  по  правому
краю!</p>
<p align="justify">А этот текст браузер
выровняет по ширине, т. е. одновременно
по правому и левому краю. Для этого бу-
дет изменена величина пробела между
словами</p>
```

Отображение в браузере:

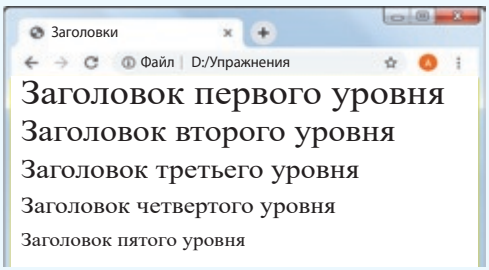


Пример 7.7. Заголовки в тексте html-документа.

Html-код:

```
<h1>Заголовок первого уровня</h1>
<h2>Заголовок второго уровня</h2>
<h3>Заголовок третьего уровня</h3>
<h4>Заголовок четвертого уровня</h4>
<h5>Заголовок пятого уровня</h5>
```

Отображение в браузере:

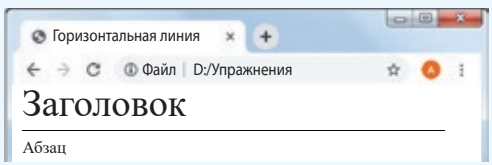


Пример 7.8. Горизонтальные линии в html-документе.

Html-код:

```
<h1>Заголовок</h1>
<hr>
<p>Абзац</p>
```

Отображение в браузере:



Пример 7.9. Способы задания цвета в html-документе.

1. По названию цвета:

```
<hr color = "red">
```

2. Шестнадцатеричным числовым кодом:

```
<hr color = "#ff0000">
```

Запомнить числовые коды цветов сложно. Для определения кода требуемого цвета можно использовать таблицы (см. *Приложение к главе 2*, с. 107) или интернет-сервисы (например, https://colorscheme.ru/).

самым мелким (пример 7.7). В тегах <h1>…<h6> аналогично тегу <p> используют атрибут align для выравнивания текста.

Иногда возникает необходимость вставить в текст html-документа перенос строки, не создавая при этом абзац. Например, при разметке стихов или текстов песен. Для этого предусмотрен непарный тег <br>, который указывает на место переноса текста.

Для разделения или дополнительного выделения блоков текста в языке HTML существует непарный тег <hr> — горизонтальная линия (пример 7.8). Горизонтальная линия растягивается на всю ширину веб-страницы и имеет один-два пикселя в толщину в зависимости от используемого браузера.

Тег <hr> поддерживает следующие атрибуты:

- align — определяет выравнивание линии;
- size — устанавливает толщину линии в пикселях;
- width — определяет ширину линии в пикселях или в процентах по отношению к ширине окна браузера;
- color — задает цвет линии.

Задавать цвет в html-документе можно двумя способами: с помощью названия (английские слова для наименования цвета) или с помощью числового шестнадцатеричного кода (пример 7.9). Перед кодом ставится знак #. В соответствии с цветовой моделью RGB две первые цифры кода задают интенсивность красного (red) цвета, третья и четвертая — зеленого (green), две последние — синего (blue).

Использование контейнера позволяет сгруппировать различные элементы веб-страницы в блок.

В примере 7.10 в блок выделено несколько абзацев. Атрибут align="center" определяет выравнивание по центру для всех абзацев, объединенных в блок.

7.4. Гиперссылки на веб-странице

Гиперссылки в html-документе создаются с помощью контейнера <a>. Внутри его могут быть: слово, группа слов или изображение.

Тег <a> поддерживает атрибуты:

- href — обязательный атрибут, который указывает на абсолютный либо относительный адрес ссылки или на имя закладки для внутренней ссылки;
- target — определяет, где будет открыт ссылаемый документ (например, _blank — в новой вкладке или окне).

Контейнер <a> может быть использован двумя способами:

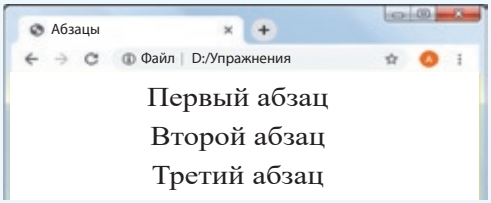
1. Для ссылки на другой документ — локальная ссылка (пример 7.11).
2. Для ссылки на элемент внутри документа — внутренняя ссылка.

Большие документы читаются лучше, если они имеют внутренние ссылки (пример 7.12). Значением атрибута href в данном случае будет так называемый якорь — ссылка на идентификатор объекта (id). Id — уникальный признак объекта, позволяющий отличать его от других объектов, т. е. идентифицировать. Имя для идентификатора лучше выбирать в соответствии с назначением ссылки.

Пример 7.10. Блоки в html-документе.
Html-код:

```
<div align = "center">
  <p>Первый абзац</p>
  <p>Второй абзац</p>
  <p>Третий абзац</p>
</div>
```

Отображение в браузере:



Пример 7.11. Локальные ссылки на веб-страницу.

1. В одном каталоге:

```
<a href = "index.html">Домой</a>
```

2. Во вложенном каталоге:

```
<a href="sait/index.html">Домой
</a>
```

3. В родительском каталоге:

```
<a href="../index.html">Домой
</a>
```

4. В соседнем каталоге:

```
<a href="../sait/index.html">
Домой</a>
```

Пример 7.12. Создание внутренней ссылки.

Создадим ссылку в конце страницы на ее начало.

Сначала создадим якорь. Первый абзац, на который будет осуществлен переход, получает идентификатор top:

```
<p id = "top"> ... </p>
```

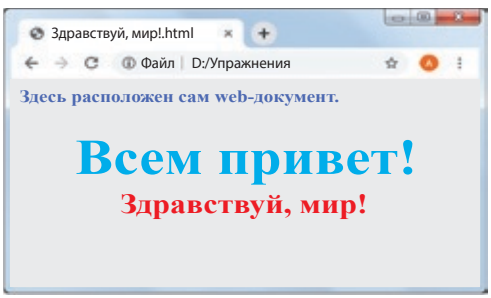
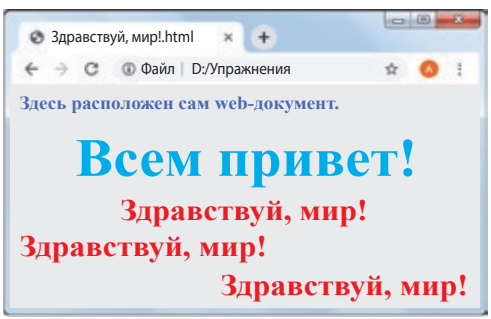
Имя ссылки в данном случае начинается с символа #:

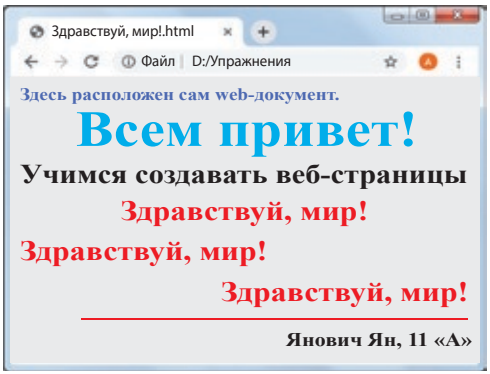
```
<a href = "#top">Наверх</a>
```

1. С помощью каких прикладных программ можно создавать веб-страницы?
2. Какое расширение имеет файл, содержащий код веб-страницы?
3. Какой элемент веб-страницы занимает ее большую часть?
4. Что такое шапка сайта? Что такое подвал сайта?
5. Какой тег используется для создания абзацев? Как отображаются абзацы в браузере?
6. Какие теги используются для создания заголовков? Какой из этих тегов определяет заголовок с самым крупным шрифтом?
7. Чем, кроме размера, отличается отображение в браузере текста заголовков и текста абзацев?
8. Какие атрибуты поддерживают теги абзацев и заголовков?
9. Как перейти на новую строку внутри абзаца веб-страницы?
10. С помощью какого тега можно разместить на веб-странице горизонтальную линию? Какие атрибуты поддерживает этот тег?
11. Какой тег определяет гиперссылку?
12. Какие ссылки используются в html-документе?

Упражнения

1 Откройте html-документ, созданный в упражнении 2 после предыдущего параграфа. Выполните указанные действия.

Действие	Результат
Измените цвет фона и основного текста (значения атрибутов `bgcolor` и `text` тега `body`). Для задания нового цвета используйте шестнадцатеричные числовые коды (#f5f5f5 — фон, #4169e0 — текст)	
Разместите на веб-странице несколько абзацев с различным выравниванием текста	

Действие	Результат
Добавьте на страницу заголовок второго уровня и горизонтальную линию. Цвет заголовка выберите самостоятельно. Под горизонтальной линией (в нижнем колонтитуле) разместите абзац с персональными данными (фамилия, имя, класс)	

2 Откройте файл index.html. Оформите абзацы этой веб-страницы как ссылки.

- 1-й абзац — ссылка на ресурс https://colorscheme.ru/
- 2-й абзац — ссылка на веб-страницу 2_2.html.

3 Добавьте на страницу 2_2.html текст «Вернуться назад» и оформите его как перекрестную ссылку на страницу index.html.

4 Создайте на странице 2_2.html внутреннюю гиперссылку. Для этого:

1. Создайте закладку в начале страницы.
2. Добавьте в конце страницы тег
 и текст «Наверх» со ссылкой на закладку.

§ 8. Понятие о каскадных таблицах стилей

В соответствии с концепцией современной веб-разработки html-код должен содержать только теги и контент в них. Для описания внешнего вида сайта используются стилевые правила специального языка разметки стилей — CSS.

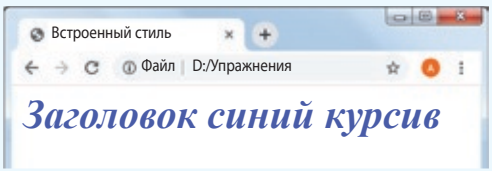
CSS (англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида html-документа.

Хокон Виум Ли (род. 27 июля 1965 г., Норвегия) — ученый, специалист в области информатики, в 1994 г. предложил использовать каскадные таблицы стилей (CSS).

Пример 8.1. Встроенные стили.
Стилевое описание:

```
<h1 style = "color: blue; font-style:
italic">Заголовок синий курсив</h1>
```

Отображение в браузере:



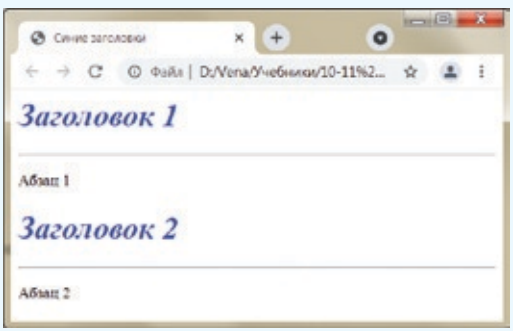
Пример 8.2. Таблицы стилей.
Стилевое описание:

```
<style>
 h1 {color: blue;
    font-style: italic}
</style>
```

Разметка веб-страницы:

```
<body>
  <h1>Заголовок 1</h1>
  <hr>
  <p>Абзац 1</p>
  <h1>Заголовок 2</h1>
  <hr>
  <p>Абзац 2</p>
</body>
```

Отображение в браузере:



CSS может быть применен как к отдельному тегу (элементу веб-страницы), так и одновременно ко всем идентичным элементам на всех страницах сайта. CSS дополняет язык HTML и значительно расширяет его возможности.

Отделяя стиль представления документов от содержимого, CSS упрощает создание веб-страниц. Отладка сайта становится более быстрой и удобной. Появляется возможность одновременной работы с одним проектом сразу нескольким разработчикам: дизайнеру, верстальщику, программисту. За счет этого повышается и скорость разработки сайта.

Способы подключения стилей CSS к html-документу:

1. Встроенные стили — стилевое описание непосредственно в открывающем теге (пример 8.1). Действует только для этого тега.

2. Таблицы стилей — стилевое описание для всех идентичных элементов веб-страницы (пример 8.2).

Задается с помощью парного тега <style>, который должен находиться в заголовке документа. Стили html-элементов внутри тега <style> задаются в соответствии с определенным синтаксисом.

3. Внешние таблицы стилей — стилевое описание html-элементов в отдельном файле (пример 8.3).

Внешние таблицы стилей — это файлы с расширением **.css**, которые содержат стилевые правила. Могут

быть созданы в любом редакторе кода. Для подключения внешних таблиц стилей в заголовок html-документа помещается тег <link> с обязательными атрибутами:

- href — ссылка на css-файл (например, href = "my_styles.css");
- rel = stylesheet — отвечает за установку взаимосвязи html-документа и css-файла;
- type = text/css — описывает тип данных в таблице стилей.

Понятие каскадности CSS заключается в том, что стили, подключенные к html-документу разными способами, имеют различные приоритеты.

Наличие приоритетов означает, что сначала будут применены параметры форматирования, заданные во внешнем css-файле, а затем — прописанные в заголовке документа. В последнюю очередь форматирование будет дополнено или изменено в соответствии с параметрами, объявленными непосредственно в тегах.

На веб-странице при помощи CSS можно изменить:

- вид, размер и цвет текстовых элементов;
- фоновые цвета и изображения;
- позицию отдельных элементов по отношению к другим;
- отступы внутри блоков и между ними;
- наличие и внешний вид обводки элементов;
- видимость или невидимость определенных фрагментов на странице.

Пример 8.2. *Продолжение.*
Синтаксис подключения стилей к html-элементам в таблицах стилей:

```
p {color: #ff0000; font-size: 14px}
```

— селектор;
— свойство CSS;
— значения свойства CSS.

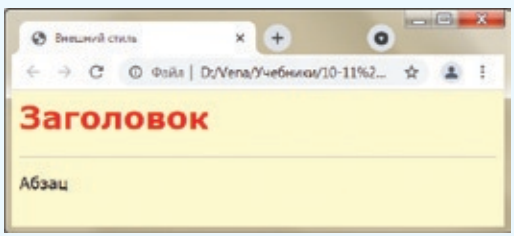
Пример 8.3. Внешние таблицы стилей.
Описание стилей в css-файле:

```
h1
{color: red}
/*Заголовок красного цвета*/
body
{background: #ffffcc; font-family:
Verdana}
/*цвет фона #ffffcc, шрифт Verdana*/
```

Подключение css-файла к html-документу:

```
<head>
  <link rel = stylesheet type = text/css
href = "1.css">
</head>
```

Отображение в браузере:



При использовании CSS следует руководствоваться следующими правилами:

1. Использовать атрибут style для какого-либо элемента, только если стиль этого элемента отличается от стилей других аналогичных элементов сайта.
2. Тег <style> со стилевым описанием использовать в том случае, если страница должна иметь индивидуальный дизайн, отличный от других страниц сайта.
3. В большинстве случаев разумно выносить стилевые описания в отдельный css-файл.

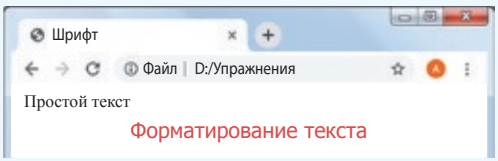
Пример 8.4. Применение стилей форматирования шрифта.

```
p {
   font-family: Tahoma;
/* Тип шрифта */
   font-weight: bold;
/* Жирность шрифта */
   font-size: 18pt;
/* Размер шрифта */
   color: tomato;
/* Цвет текста */
   text-align: center;
/* Выравнивание текста */
  }
```

Сокращенная запись:

```
p {
   font: bold 18pt Tahoma;
color:tomato; text-align:center;
  }
```

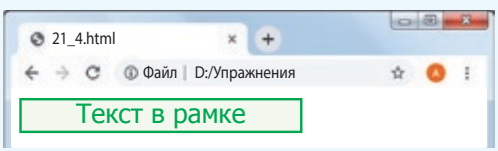
Отображение в браузере:



Пример 8.5. Применение стилей для блока.

```
div {
   width:80%;
/*Ширина блока */
   background: #f0fff0;
/*Цвет фона*/
   border: 3px solid limegreen;
/*Толщина, стиль и цвет границы */
   padding: 10px; /*Поля*/
   font: 18pt Tahoma;
/*Размер и тип шрифта */
   color: seagreen;
/*Цвет текста */
   }
```

Отображение в браузере:



С помощью CSS можно определять стиль и вид текста аналогично использованию тега <font>, задающего свойства шрифта. При этом стили обладают большими возможностями и позволяют сократить html-код. С атрибутами CSS, задающими свойства шрифта, можно познакомиться в *Приложении к главе 2* (с. 110).

Стили шрифта задаются большим перечнем отдельных свойств. Запись, определяющую свойства шрифта, можно сократить. В сокращенной записи нужно быть осторожным с порядком указания свойств. Например, свойство `size` должно задаваться раньше, чем свойство `family` (пример 8.4).

Кроме изменения параметров шрифтов, можно управлять и свойствами всего текста. Значения свойств приведены в *Приложении к главе 2* (с. 110).

С помощью CSS задаются свойства блоков (пример 8.5).

Внутренние отступы в блоке (отступы от внешней границы блока до его содержания) задает свойство `padding`. Эти отступы иногда называют полями.

Существует несколько способов задания полей:

1) `padding: 10 px` — одинаковые отступы со всех сторон;

2) `padding: 5 px 10 px` — отступы сверху и снизу 5px, справа и слева 10px;

3) `padding: 5 px 10 px 15 px` — отступы сверху 5 px, слева и справа 10 px, снизу 15 px;

4) `padding: 5 px 10 px 15 px 20 px` — разные отступы со всех сторон, слева направо: верхний, правый, нижний, левый.

Внешние отступы блока (отступы от внешней границы блока до границ страницы или до соседних элементов) задает свойство `margin`. Способы задания внешних отступов аналогичны способам задания внутренних (пример 8.6).

Когда необходимо определить стиль для отдельного элемента веб-страницы или задать разные стили для одного тега, применяют классы (пример 8.7).

Синтаксис при использовании классов:

```
.Имя_класса {
свойство1: значение; свойство2:
значение; ... }
```

Чтобы указать в html-коде, что тег используется с определенным классом, к тегу добавляется атрибут `class="Имя_класса"`.

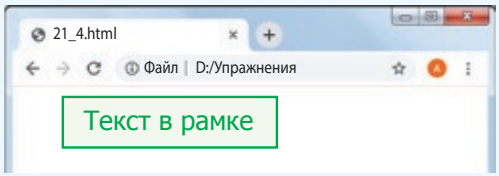
Классы гибкие, их можно создавать много и называть понятными именами. Имя класса может содержать в себе латинские буквы, цифры, символ дефиса (-) и подчеркивания (_). Начинаться имя должно с латинской буквы.

Созданный класс можно применять к любым элементам веб-страницы. С помощью классов можно придавать стиль не только целым заголовкам и абзацам, но и отдельным фрагментам страницы, например словам. Для этого нужно заключить слово в контейнер `<span>` и добавить атрибут `class`.

Пример 8.6. Отступы в блоке.

```
padding: 20px 10px;
margin: 10px 80px;
```

Отображение в браузере:



Пример 8.7. Использование классов в CSS.

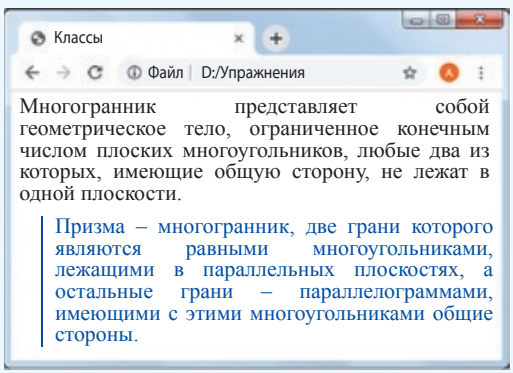
Стилевое описание:

```
p {/* Обычный абзац */
   text-align: justify;
/*Выравнивание текста по ширине*/
   }
  .figura {/* Абзац с классом figura */
   color: navy;
/*Цвет текста*/
   margin-left: 20px;
/*Отступ слева*/
   border-left: 1px solid navy;
/*Граница слева от текста*/
   padding-left: 15px; /*Расстояние от
линии до текста*/
   }
```

Указание на использование класса:

```
<p class = "figura">
```

Отображение в браузере:



1. Что такое каскадные таблицы стилей?
2. Какими способами можно подключить каскадные таблицы стилей к html-документу?
3. Каким образом определяются приоритеты при подключении каскадных таблиц стилей различными способами к одному html-документу?
4. Что можно изменять на веб-странице с помощью каскадных таблиц стилей?

Упражнения

1 Откройте файл 8.html. Оформите абзацы этой веб-страницы, используя встроенные стили.

- 1-й абзац — style = ''color: #c71585; font-style: italic'' (цвет текста — лилово-красный, начертание — курсив).
- 2-й абзац — style = ''color: #c71585; font-size: 16px; text-align: justify'' (цвет текста — лилово-красный, размер шрифта — 16 px, выравнивание текста — по ширине).
- 3-й абзац — style = ''color: #dc143c; font-family: Arial'' (цвет текста — малиновый, шрифт — Arial).
- 4-й абзац — style = ''color: teal; font-family: Verdana; font-size: 16px'' (цвет текста — зеленовато-синий, шрифт — Verdana, размер шрифта — 16 px).

Сохраните страницу под новым именем.

2 Откройте файл 8.html. Оформите веб-страницу, используя таблицы стилей.

```
<style>
p {color: teal; font-style: italic; text-align: justify}
  h1 {font-family: Verdana; text-align: center}
  </style>
```

Сохраните страницу под новым именем.

3 Откройте файл 8.html. Оформите веб-страницу, используя внешние таблицы стилей (описание стилей в файле). Параметры стилей задайте самостоятельно. Сохраните страницу под новым именем.

4 Откройте файл. Примените к блокам текста различное форматирование, используя классы.

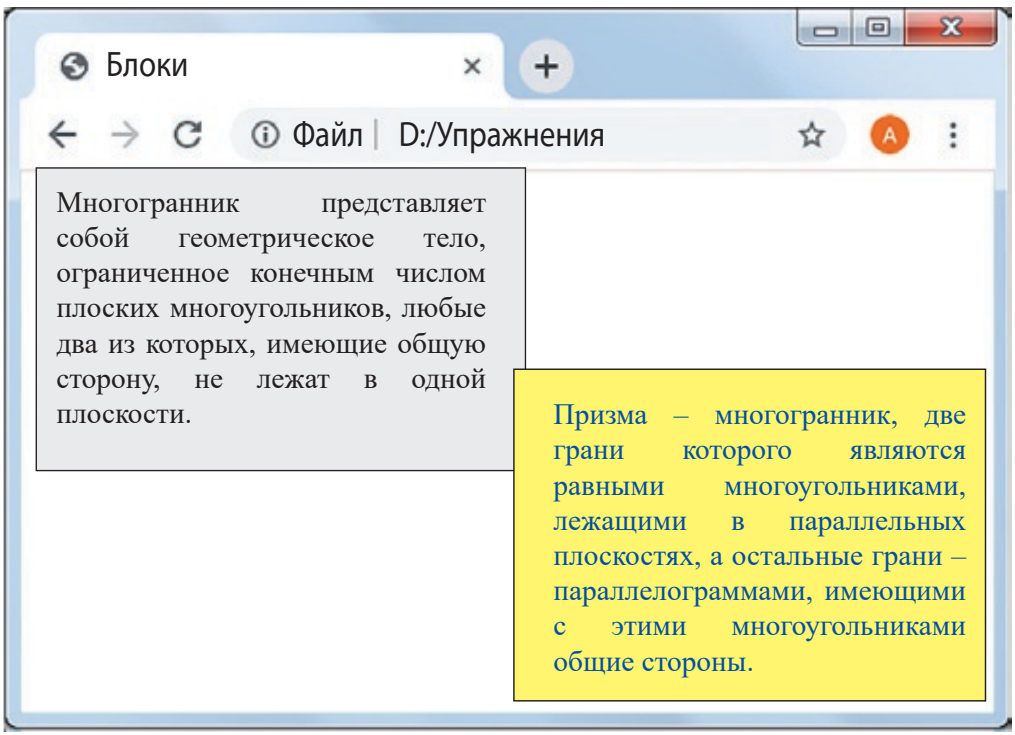
1-й блок:

```
div.b1 {
width: 300px; /* Ширина */
background: #d3d3d3; /* Цвет фона */
padding: 8px; /* Поля */
padding-right: 40px; /* Правое поле */
border: solid 1px black; /* Стиль, толщина и цвет границы */
float: left; /*Выравнивание блока по левому краю */
}
```

2-й блок:

```
div.b2 {
width: 300px;
background: #f0e68c;
padding: 8px;
border: solid 2px black;
float: left;
position: relative; /* Расположение блока относительно первого блока */
top: 40px; /* Смещение по вертикали вниз */
left: -70px; /*Смещение по горизонтали влево*/
}
```

Должно получиться:



Используйте класс для форматирования слова *Призма* во втором блоке.
Описание класса:

```
.termin {font-family: Verdana}
```

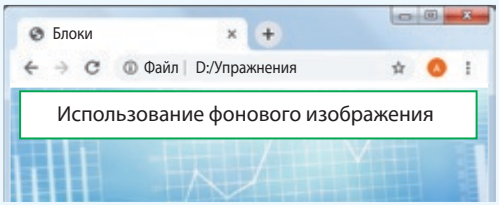
Применение класса:

```
<span class = "termin">Призма</span>
```

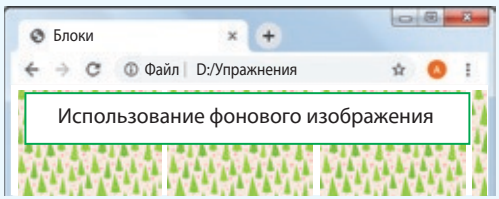
§ 9. Мультимедиа на веб-страницах

Пример 9.1. Использование фонового изображения.

Размер изображения 1920 × 1200 px:



Размер изображения 225 × 225 px:



Пример 9.2. Разметка фонового изображения.

Файл с изображением находится в одной папке с html-файлом:

```
<body background = "fon.jpg">
```

Файл с изображением находится в папке с именем img:

```
<body background="img/fon.jpg">
```

Пример 9.3. Добавление свойства width к изображению.

```
img {
  width: 300px;
/* Ширина в пикселях */
  }
```

При использовании процентной записи ширина изображения определяется в зависимости от ширины родительского элемента (например, блока). Если родительский элемент не указан, то в его качестве выступает окно браузера.

```
img {
  width: 80 %;
/* Ширина в процентах */
  }
```

9.1. Графика на веб-страницах

Использование изображений на веб-страницах позволяет наглядно донести до аудитории нужную информацию. Графика дает возможность немного оживить общее оформление веб-страницы. Кроме того, для оформления веб-страниц применяется деловая графика, т. е. различные графики, диаграммы и схемы.

На веб-страницах изображения активно используются не только в качестве иллюстраций к тексту, но и для фонового оформления (примеры 9.1).

Если фоновое изображение меньше размера веб-страницы, оно будет размножено по горизонтали и вертикали.

При использовании изображения в качестве фона необходимо помнить, что оно не должно отвлекать внимание от текста и должно хорошо сочетаться с цветовой гаммой веб-страницы.

Файлы с изображениями, используемыми на страницах сайта, принято хранить в отдельной папке.

Чтобы поместить изображение на веб-страницу как фон, нужно установить в теге <body> атрибут background со значением, содержащим полное имя файла с изображением (пример 9.2).

Для добавления иллюстрации на страницу используется непарный тег источника изображения:

<img src="полное имя файла">,

где src – обязательный атрибут тега img.

Тег <img> предназначен для отображения на веб-странице изображений графических форматов GIF, JPEG, PNG или SVG.

Основные свойства CSS, применяемые к тегу <img>:

- width — ширина изображения;
- height — высота изображения;
- padding — отступы от изображения;
- float — выравнивание изображения;
- border — рамка вокруг изображения.

Свойства width и height используются для изменения размеров изображения (пример 9.3). Значения этих свойств можно задавать в пикселах или процентах. Добавление только одного из этих свойств позволит сохранить пропорции изображения.

По умолчанию текст на веб-странице располагается вплотную к изображению, что нарушает эстетическое восприятие контента. Этого можно избежать, задав для изображения невидимые отступы по горизонтали (слева и справа) и вертикали (сверху и снизу). Отступы для изображений задаются так же, как для блоков, — значениями в пикселях для свойства padding (пример 9.4).

Добавить на страницу изображения так, чтобы они полностью обтекались текстом, можно с помощью стилевого свойства float. Значение right будет выравнивать изображение по правому краю окна браузера или блока, а left — по левому краю. Обтекание при этом происходит по другим, свободным сторонам. В примере 9.5 изображение и текст размещаются в блоке.

Пример 9.4. Отступы для изображения[1].

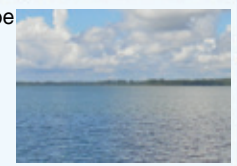
```
img {
  padding: 10px;
/* Одинаковые отступы со всех сторон */
  }
```

Пример 9.5. Выравнивание изображения по отношению к тексту.

```
img {
float: left;
/* Изображение слева от текста */
padding:  5px;
/* Отступ */
}
```

Нарочь – самое большое озеро в Беларуси. Площадь его зеркала – 79,6 кв. км. Наибольшая глубина – 24,8 км. Длина береговой линии – 40 км (четвертое место среди озер Беларуси). Объем воды – 710 млн кв. м.

```
img {
   float: right; /* Изображение справа
от текста */
   }
```



Нарочь – самое большое озеро в Беларуси. Площадь его зеркала – 79,6 кв. км. Наибольшая глубина – 24,8 км. Длина береговой линии – 40 км (четвертое место среди озер Беларуси). Объем воды – 710 млн кв. м.

[1] Текст в примере цитируется по: https://gp.by/category/news/culture/news21091.html (дата доступа: 28.08.2019).

Пример 9.6. Альтернативный текст.

```
<img src = "ozero.jpg" alt = "Нарач">
```

Максім Танк
Люблю твае, Нарач, затокі і тоні,
Як вецер густыя туманы развесіць.
Ці снежная пена на хвалях зазвоніць,
Цалуючы зоры, калышучы месяц.

Пример 9.7. Использование изображения в качестве ссылки.

Стилевое описание:

```
img {border: none}
```

Html-код:

```
<a href = "sample.html">
  <img src = "link.png" alt = "Пример">
  </a>
```

Атрибутов, которые могут быть использованы в теге <img>, и свойств, используемых при стилевом описании, большое количество. С ними можно познакомиться на ресурсе http://htmlbook.ru/content.

При включении мультимедийных элементов на страницу необходимо хорошо представлять, будут ли оправданы затраты времени, идущие на загрузку. Мультимедийные файлы занимают много памяти. Самые большие из них — видеофайлы, минутный фрагмент может занимать более 10 Мбайт памяти. Звуковые файлы требуют немного меньше места — трехминутная запись высокого качества может занимать до 3,5 Мбайт.

Для добавления звука на веб-страницы предпочтительнее использовать файлы с расширением .mp3. Данные в таких файлах упакованы с учетом особенностей восприятия человеческим ухом.

К тегу <img> можно применять атрибут alt, который устанавливает альтернативный текст для изображений. Такой текст позволяет получить текстовую информацию о рисунке при отключенной в браузере загрузке изображений (пример 9.6).

Изображение на веб-странице можно сделать ссылкой на другой файл, поместив тег <img> в контейнер <a>. Если при этом вокруг изображения отображается рамка, то убрать ее можно, определив значение none для стилевого свойства border (пример 9.7).

9.2. Звук и видео на веб-страницах

В настоящее время, используя Интернет, можно передавать не только текстовые данные и изображения, но и видео, звук и все то, что называется мультимедиа.

Под термином **мультимедиа** понимают компьютерное представление информации, состоящее из более чем одного типа данных, к которым можно отнести текст и звук, видео и звук.

Звук на веб-странице можно использовать как один из декоративных компонентов. Однако встречаются веб-сайты, на которых он превращается из декоративного компонента в основной (сайты музыкальной тематики, музыкальные архивы).

Контейнер <audio> добавляет, воспроизводит и управляет настройками аудиозаписи на веб-странице. Путь

к файлу задается через атрибут src (пример 9.8).

Атрибуты тега <audio>:

- src — указывает путь к воспроизводимому файлу;
- autoplay — звук воспроизводится сразу после загрузки страницы;
- controls — добавляет панель управления звуком;
- loop — повторяет воспроизведение звука после его завершения;
- preload — используется для загрузки файла вместе с загрузкой веб-страницы.

Управление воспроизведением звука различается между браузерами, но основные элементы совпадают. Внутри контейнера <audio> может располагаться поясняющий текст, который будет выводиться в браузерах, не работающих с этим тегом.

Контейнер <video> добавляет, воспроизводит видеоролик и управляет его настройками на веб-странице. Путь к файлу задается так же, как и при использовании звука (пример 9.9).

Атрибуты тега <video>:

- src — указывает путь к воспроизводимому файлу;
- autoplay — видео начинает воспроизводиться автоматически после загрузки страницы;
- controls — добавляет панель управления к видеоролику;
- loop — повторяет воспроизведение видео после его завершения;
- height — задает высоту области для воспроизведения;

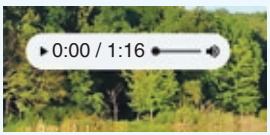
При добавлении видео на веб-страницы следует обратить внимание на файлы с расширениями .webm и .mp4.

MP4 — формат, позволяющий хранить видео высокого качества и оптимальной портативности с точки зрения хранения файлов.

WebM — открытый формат для мультимедиа-файлов, представленный компанией Google. Не требует лицензионных соглашений.

Пример 9.8. Добавление звука.

```
<audio controls width = "190px" src =
"muz.mp3">
Тег audio не поддерживается вашим
браузером.
</audio>
```



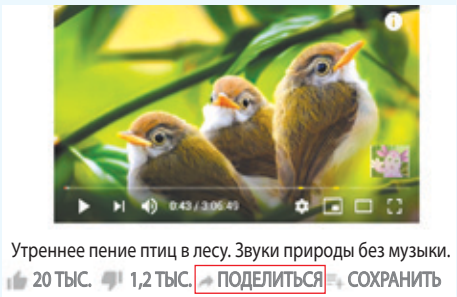
Пример 9.9. Добавление видеоролика.

```
<video controls width = "190px" src =
= "vid.mp4">
</video>
```

Для универсального воспроизведения в различных браузерах аудио и видео кодируют с помощью разных кодеков и одновременно во вложенном теге <source> добавляют ссылки на файлы разных форматов:

```
<video width = "400" controls =
 "controls">
 <source src = "duel.mp4" type= 'video/
 mp4; codecs = "avc1.42E01E, mp4a.40.2"'>
 <source src= "video/duel.webm" type =
 'video/webm; codecs = "vp8, vorbis"'>
</video>
```

Пример 9.10. Добавление видеоролика с сервиса YouTube.



- `width` — задает ширину области для воспроизведения;
- `preload` — используется для загрузки видео вместе с загрузкой веб-страницы.

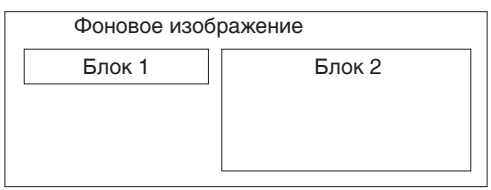
Необходимо всегда включать в тег `<video>` атрибуты ширины и/или высоты. Если значения этих атрибутов не заданы, браузеру не известен размер видео. В этом случае страница начнет обновляться при загрузке видео, и будет создаваться эффект мерцания.

Любое видео, размещенное на веб-сервисе YouTube, также можно добавить на создаваемую веб-страницу. Для этого нужно нажать кнопку **Поделиться** и выбрать **Встроить**, а затем скопировать полученный html-код в любое место html-документа (пример 9.10).

1. В каком качестве можно использовать графику на веб-страницах?
2. Каким должно быть изображение, используемое как фон веб-страницы?
3. В каком теге и с помощью какого атрибута на веб-страницу добавляется фоновое изображение?
4. Какой тег позволяет добавить на веб-страницу иллюстрацию?
5. Каким образом определяется положение иллюстрации на веб-странице?
6. Для чего предназначен атрибут `alt`?
7. Как сделать изображение ссылкой?
8. С помощью каких тегов на веб-страницу добавляется звук и видео?
9. Как на веб-страницу добавить видео с веб-сервиса?

Упражнения

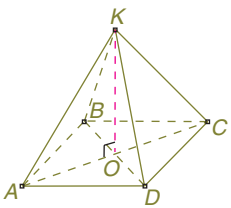
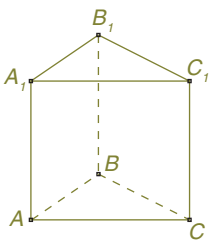
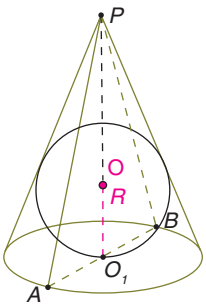
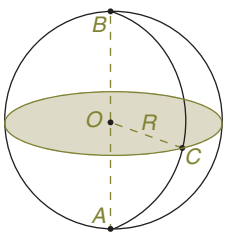
1 Создайте веб-страницу следующей структуры:



Блок 1:

Отображение в браузере	Стилевое описание
Стереометрия	`div.bl1 { width: 300px; float: left; padding: 15px; margin: 30px; font: 30pt Tahoma; color: #808000; }`

Блок 2:

Отображение в браузере	Стилевое описание
Пирамида  Призма  Конус  Сфера 	`div.bl2 { width: 600px; background: #ffffff; padding: 15px; margin: 30px; border: solid 1px black; font: 14pt Tahoma; float: left; }`

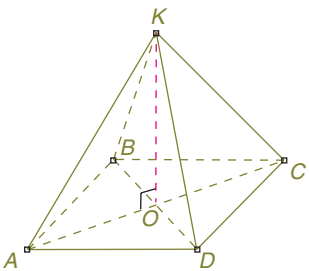
Сохраните результат как файл index.html. Все рисунки, используемые на веб-странице, сохраняйте в отдельной папке.

2 Для каждой фигуры из упражнения 1 создайте веб-страницу следующей структуры:

Элементы страниц:

Элемент	Пример отображения в браузере	Стилевое описание

Заголовок	ПИРАМИДА	`h1{`
		`  text-align: center;`
		`  font-family: Comic Sans MS;`
		`  color: #DC143C;`
		`  }`

Блок	 Пирамида — многогранник, одна из граней которого (называемая основанием) — произвольный многоугольник, а остальные грани (называемые боковыми гранями) — треугольники, имеющие общую вершину.	`div {`
		`  width: 620px;`
		`  height: 300px;`
		`  background: #ffffff;`
		`  border: solid 1px black;`
		`  padding: 15px;`
		`  margin: 0  auto;`
		`  font: 14pt Arial;`
		`  line-height: 1.5;/`
		`*Межстрочный  интервал*/`
		`  }`

Абзац	Главная Следующая	`p {`
		`  font: 14pt Tahoma;`
		`  text-align: center;`
		`  }`

Элемент	Пример отображения в браузере	Стилевое описание

Линия		hr {
		color: #DC143C;
		background-color: #DC143C;
		/*Цвет линии для разных браузеров*/
		height: 2px;
		width:700px;
		}

Стилевое описание выполните в отдельном файле. Подключите стилевой файл к каждой странице. Сохраните страницы как 1.html, 2.html, 3.html, 4.html.

3 Свяжите страницу index.html и страницы, созданные в упражнении 2, с помощью перекрестных гиперссылок. В качестве ссылок на странице index.html используйте изображения фигур, а на страницах второго уровня — текстовые элементы абзаца.

4 Добавьте на страницу index.html звук (воспроизведение при загрузке).

5 На одну из страниц упражнения 2 добавьте видеоролик соответствующей тематики. Для поиска видеоролика воспользуйтесь веб-сервисом YouTube.

§ 10. Визуальное веб-конструирование

Визуальный метод позволяет автоматизировать процесс конструирования веб-сайтов. Этот метод возник из необходимости уменьшить трудоемкость и время создания сайтов.

Суть метода отражена в принципе WYSIWYG (от англ. *What you see is what you get* — «Что видишь, то и получишь»). При создании сайтов с использованием визуального метода необходимо выбрать инструменты: редакторы визуального конструирования, или онлайн-конструкторы сайтов. В § 6 упоминались некоторые из них. Использование редакторов обеспечивают пользователю возможность работать без непосредственного подключения к Интернету (пример 10.1).

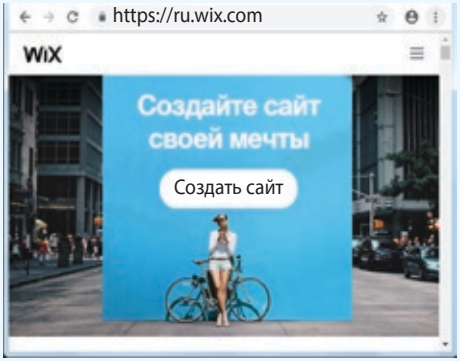
Пример 10.1. Визуальный веб-редактор Adobe Dreamweaver.

С возможностями этого редактора можно познакомиться, перейдя по ссылке:

https://www.adobe.com/products/dreamweaver.html

Dreamweaver не только генерирует html-код, но также предоставляет различные полезные инструменты, такие как библиотеки фрагментов кода, управление файлами, сервер отладки и др. Используя этот редактор, можно просматривать CSS информацию в единой унифицированной CSS-панели, что позволяет легко увидеть стили применительно к конкретным элементам и редактировать существующие стили.

Пример 10.2. Конструкторы сайтов.
1. Wix — https://ru.wix.com/

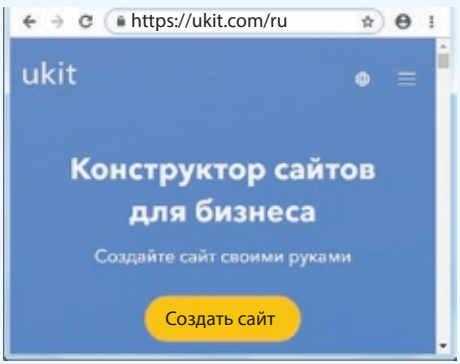


Один из самых популярных конструкторов сайтов. Он позволяет создавать ресурсы любого формата, начиная с блогов и визиток и заканчивая онлайн-магазинами. На выбор пользователей предлагается свыше 500 шаблонов.

Сервис отличается интуитивно понятным интерфейсом, но имеет и недостатки:

- два наиболее доступных тарифных плана из пяти предлагают ограниченную функциональность;
- самый дешевый тарифный план, как и бесплатный, не позволяет удалить рекламу Wix;
- нельзя выбрать короткий и запоминающийся домен.

2. Ukit — https://ukit.com/ru



В дальнейшем сайт можно разместить в Интернете. При использовании редакторов для создания веб-сайтов html-код генерируется в соответствии с визуальным представлением веб-страницы.

В современном мире любая организация должна иметь свой сайт, иначе она становится неконкурентоспособной. В связи с этим каждый день по всему миру появляются тысячи новых сайтов. И, как следствие, активно развивается индустрия шаблонных сайтов на конструкторах.

Конструкторы веб-сайтов представляют собой законченные решения для создания и последующего управления веб-сайтами любой сложности и назначения. Веб-конструкторы позволяют автоматизировать процессы создания не только гипертекста, но и структуры и дизайна веб-сайта.

Некоторые конструкторы сайтов предлагают комплексные решения для построения сайтов различной направленности «под ключ», при этом не требуется устанавливать никакое программное обеспечение, нужен только доступ в Интернет (пример 10.2).

Преимущества конструкторов сайтов:

1. Цена. При использовании некоторых конструкторов можно создать сайт бесплатно, а потом загрузить его в Интернет на хостинг.

2. Скорость создания. Сайт можно создать за несколько часов.

3. Скорость работы. Сайты уже оптимизированы под площадки, на которых они находятся, и будут быстро работать.

4. Отсутствие программирования. Для создания сайта не нужно знать ни языков веб-программирования, ни языков разметки.

5. Готовая структура сайта. При создании сайта предоставляется уже готовая структура с примерами.

Исходя из всего вышесказанного, конструктор сайтов может быть эффективным инструментом для тех, кто ограничен во времени, или для тех, кому нужен простой сайт, а также в случае, когда нет уверенности в дизайнерских способностях.

Пример 10.2. *Продолжение.*

Конструктор Ukit ориентирован на небольшие фирмы, которые без лишних затрат и усилий хотят получить красивый и удобный сайт. Сервис предлагает несколько сотен шаблонов.

Бесплатно пользоваться конструктором можно лишь в рамках пробного периода, после которого нужно подключить один из тарифов. Самый доступный из них уже включает в себя домен вида site.ru, неограниченное количество страниц и места на сервере.

Пробный период составляет 14 дней.

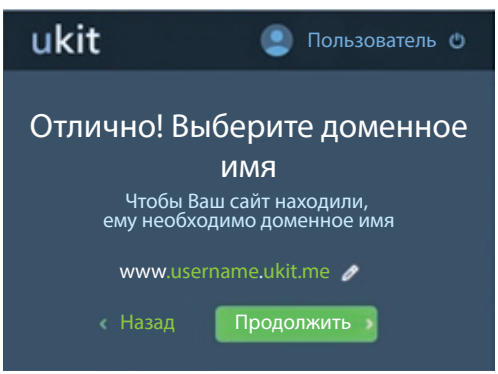
1. Что представляет собой метод визуального веб-конструирования?
2. Какие инструменты создания сайтов используются при визуальном методе веб-конструирования?
3. Какими преимуществами обладают конструкторы сайтов?

Упражнения

1 Зарегистрируйтесь на сайте http://ukit.com[1] и подтвердите регистрацию через ссылку в электронном письме.

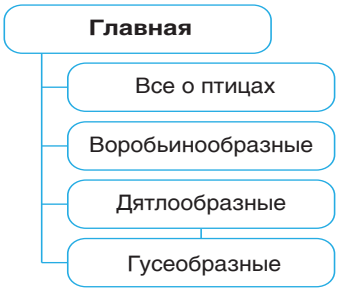
2 Используя предварительный просмотр, рассмотрите предлагаемые шаблоны. Выберите шаблон для будущего сайта из категории «Универсальные».

3 Введите доменное имя сайта, изменив username (например, *birds*).



[1] Все упражнения после § 22 можно выполнить с помощью другого конструктора сайтов, например https://ru.wix.com/

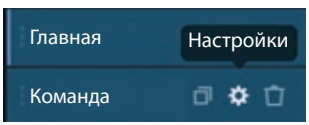
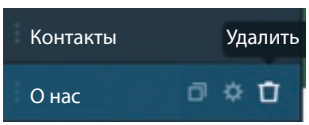
4 На основе выбранного шаблона создайте сайт следующей структуры:



Для этого:

1. Перейдите на панель **Страницы сайта**  .

2. В разделе **Основные** переименуйте страницы. Лишние страницы удалите. При необходимости создайте новые.

Переименование	Удаление
	

3. Поменяйте порядок страниц (если необходимо, страницы можно перетаскивать).

4. Поменяйте названия страниц в меню (для открытия панели **Меню** просто выделите меню на странице).

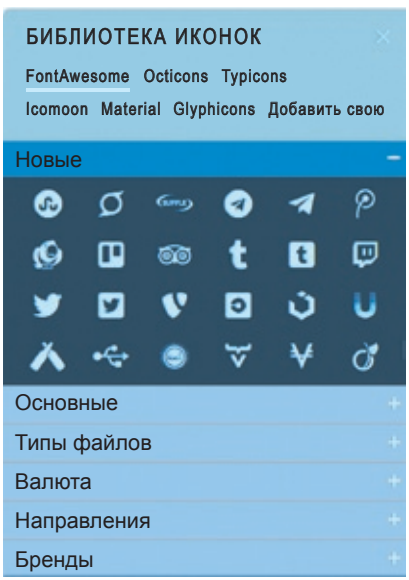
5 В шапке сайта разместите логотип с иконкой, соответствующей тематике сайта. Для замены иконки выполните перечисленные действия.

5.1. Выполните щелчок по имеющейся в шаблоне иконке.

5.2. Нажмите **Заменить**.

5.3. В окне **Библиотека** щелкните по подходящей иконке.

5.4. Настройте цвет и размер иконки.



6 В нижний колонтитул сайта добавьте персональные данные (фамилию, имя, класс).

7 Измените содержание страниц в соответствии с тематикой сайта (блоки можно удалять/добавлять, размещать в них свой текст, рисунки и другие объекты). Материалы для оформления страниц можно взять с сайта http://e-vedy.adu.by.

8 На странице «О птицах» разместите:

1. Заголовок 1 <Название страницы>.
2. Текст о птицах.
3. Слайдер <Фото птиц>.

9 На страницах «Воробьиные» и «Дятлообразные» разместите:

1. Заголовок 1 <Название страницы>.
2. Контент <Название, фото и описание птицы> — в 3 экземплярах.

10 Найдите в Интернете информацию для страницы «Гусеобразные». Оформите ее аналогично страницам «Воробьинообразные» и «Дятлообразные».

11 Страницу «Главная» оформите самостоятельно (продумайте содержание и дизайн).

§ 11. Разработка фрагментов тематических сайтов

Используя для поиска информации Интернет, вы обычно посещаете десятки разнообразных сайтов. Многие из них привлекают внимание своим внешним оформлением, удобно размещенной информацией, хорошо продуманными средствами навигации. Но есть и такие, которые вызывают раздражение долгой загрузкой или неудачно выбранным дизайном, мешающим воспринимать информацию, а порой просто не содержат никакой полезной информации. Посещать еще раз такие сайты желания не возникает. Поэтому, приступая к разработке собственного сайта, следует тщательно продумывать многие вопросы.

Рассмотрим этапы разработки вебсайта (пример 11.1).

1. Проектирование сайта — постановка целей и задач сайта.

Определите, информацию какого типа вы хотите разместить на страницах

Пример 11.1. Создание сайта «Поэтические образы в русской литературе».

Этапы разработки

1. Разработать сайт по учебному предмету «Русская литература», 11-й класс.
2. Структура сайта.

Изображения на главной странице будут являться гиперссылками на страницы второго уровня. На каждой странице второго уровня будут ссылки на соседнюю и главную страницу.

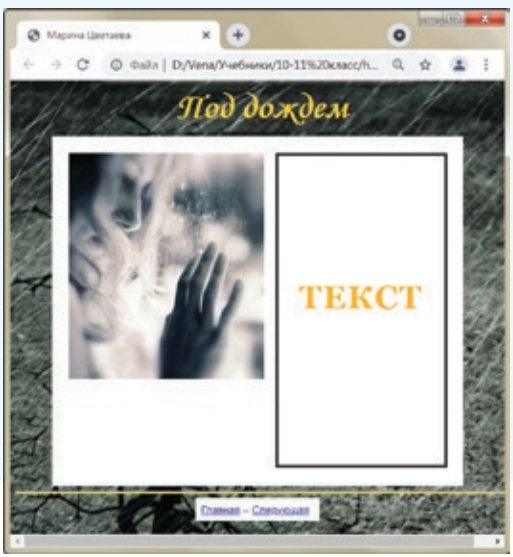
3. Дизайн сайта.

Главная страница. Содержит заголовок и блок с изображениями и персональными данными русских поэтов начала XX в.

Пример 11.1. *Продолжение.*

Фоновое изображение страницы — имитация бумаги. Фон блока — сплошной цвет пастельного оттенка.

Страницы второго уровня. Содержат заголовок, блок с иллюстрацией и текстом стихотворения, горизонтальную линию и блок с элементами навигации. Цветовое решение стремится к минимализму. Фоновое изображение и иллюстрация различаются для каждой из пяти страниц.



сайта; для какой аудитории и возрастной группы предназначен сайт. От этого будет зависеть и внешнее оформление, и форма подачи материала, и организация навигации.

2. Разработка структуры сайта.

Не стоит размещать на одной странице слишком большой объем информации, т. к. это мешает ее восприятию. Лучше создать несколько связанных страниц, разместив на них информацию, сгруппированную по тематике.

Навигацию сайта нужно тщательно продумать. Основные разделы сайта должны быть доступны с главной страницы. На каждой странице необходима ссылка, с помощью которой посетитель сможет вернуться на главную страницу.

3. Создание дизайна сайта (внешнее оформление сайта имеет немаловажное значение для его посещаемости).

Преимущество имеют светлые и контрастные цветовые решения (при выборе цветового оформления можно воспользоваться рекомендациями профессионалов).

Не следует перегружать сайт большим количеством графической информации, замедляющей его загрузку.

Дизайн главной страницы создается отдельно и отличается от дизайна типовых страниц.

На главной странице не должно быть текста большого объема.

4. Создание мультимедиа-элементов.

5. Верстка страниц и шаблонов (создание html-кода).

6. Программирование (в случае разработки сложного, многофункционального сайта).

7. Наполнение контентом.

8. Тестирование и внесение корректировок.

9. Публикация сайта на хостинге.

10. Обслуживание работающего сайта или его программной основы.

В зависимости от поставленной задачи некоторые этапы могут отсутствовать.

Создание сайта является довольно сложной работой. От каждого этапа и безошибочной его реализации зависит качество всего сайта.

Пример 11.1. *Продолжение.*

4. На каждой странице второго уровня присутствует соответствующее звуковое сопровождение.

5. При верстке страниц используется ручной метод и редактор html-кода. Подключаем таблицу стилей для главной страницы. Для страниц второго уровня создаем и подключаем файл с описанием стилей.

6. Сайт содержит только статические страницы. Для их создания не требуется использовать языки веб-программирования.

7. Для поиска текстов стихотворений и изображений поэтов используем Интернет.

8. Проводим поэтапное (для каждой страницы) и окончательное (проверяем функциональность навигации) тестирование.

9—10. Цель создания сайта — обучение. Хостинг и обслуживание не производятся.

1. Какие этапы можно выделить в процессе создания сайта?
2. Почему при разработке сайта необходимо следовать определенным этапам?
3. В чем суть каждого этапа создания сайта?

Упражнения

1 Дополните сайт из примера 11.1 недостающими страницами.

2 Создайте сайт (можно использовать образец примера). Предлагаемая тематика:

1. Крупнейшие трансконтинентальные компании.
2. Ведущие промышленные предприятия нашей страны.
3. Модели кристаллических решеток.
4. Электрический ток в различных средах.
5. Мифологические основы названия созвездий.
6. Тригонометрические функции.
7. Система образования Республики Беларусь.

Глава 3
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБЩЕСТВЕ

§ 12. Информационные системы, технологии и ресурсы

Пример 12.1. Разнообразие систем.

Системы в природе: солнечная система, живой организм, растение и др.

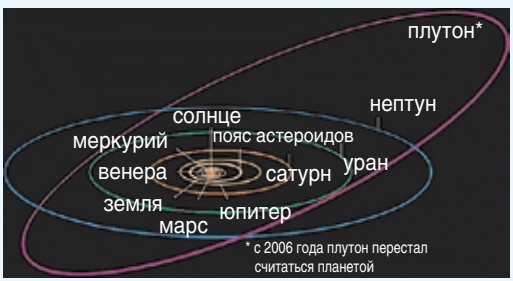
Технические системы: компьютер, компьютерная сеть, электростанция, автомобиль и др.

Социальные системы: школа, социальная сеть, город и др.

Нематериальные системы: разговорный язык, язык химических формул, нотная запись, система программирования и др.

Пример 12.2. Структура системы.

Структура солнечной системы:



Структура компьютерной сети:

12.1. Информационные системы

Реальные объекты, окружающие человека, очень сложные, поэтому их принято рассматривать как систему. Система состоит из элементов, которые связаны между собой. Все вместе они выполняют общие функции, что позволяет рассматривать их как единое целое (пример 12.1).

Совокупность выделенных отношений (взаимосвязей) между элементами системы принято называть **структурой** системы. Отличительной особенностью системы является наличие у нее таких качеств или функций, которые не свойственны ни одному ее элементу, ни одной ее подсистеме, взятым в отдельности. Структура системы может изменяться в результате протекания каких-либо процессов: элементы системы могут добавляться или удаляться. Удаление элемента системы или появление нового всегда приводят к изменению взаимосвязей между элементами (пример 12.2).

Система характеризуется функциями, назначением, входами и выходами, внутренним состоянием. Система оценивается определенным набором качественных и количественных показателей — параметров системы.

Понятия «информация», «информационный процесс», «информационная система» взаимосвязаны. Информация проявляется в информационных процессах, которые протекают только

в рамках какой-либо информационной системы.

Информационная система — это система, элементами которой являются данные, технические средства, методы и специалисты, а связи образуются благодаря протеканию информационных процессов. Наличие информации (данных) и информационных процессов позволяет говорить об информационной системе (пример 12.3).

Информатика изучает закономерности протекания информационных процессов в системах различной природы, но в наибольшей степени предметом ее исследований являются информационные процессы в технических и социотехнических системах. Причем эти закономерности важны с точки зрения возможности автоматизации этих процессов. Под **автоматизированной информационной системой** (АИС) понимается совокупность информационных массивов, технических, программных и языковых средств и персонала, предназначенных для сбора, хранения, поиска, обработки и выдачи данных по запросам пользователей.

Автоматизированные информационные системы применяются практически во всех сферах человеческой деятельности: в управлении производством или предприятием; при организации научных исследований и выполнении конструкторских и проектных работ; в библиотечном деле; в обучении и т. д.

В примере 12.4 представлены автоматизированные информационные системы разных видов.

Пример 12.3. Информационная система.

Пример 12.4. Виды АИС:

• *информационно-справочные* (ИСС) — разнообразные электронные словари, электронные энциклопедии, электронные записные книжки и т. д.;

• *информационно-поисковые системы* (ИПС) — наиболее известными являются всемирная паутина (WWW) с соответствующими поисковыми системами (Google, Yandex и др.) и юридические ИПС, предназначенные преимущественно для хранения документов официального характера (законов, положений, инструктивных писем и др.), изданных государственными органами;

• *геоинформационные системы* (ГИС). В них информация об объектах упорядочена в соответствии с пространственным размещением объектов, представленных чаще всего на географических картах;

• *измерительные ИС* используются для автоматического (с помощью специальных датчиков) сбора информации о состоянии и параметрах интересующего объекта. Применяются измерительные АИС в медицине, метеорологии, сейсмологии, при организации космических полетов, на атомных электростанциях, на производствах, вредных для здоровья человека, и т. д.;

• *системы автоматизации научных исследований* снабжены средствами для построения информационных моделей самого разного вида;

Пример 12.4. *Продолжение.*

- *экспертные системы* (ЭС) и *системы поддержки принятия решений* (СППР). Их основу составляют базы знаний (БЗ) по конкретной предметной области. Данные системы активно используются при планировании и составлении долгосрочных прогнозов в промышленности, для постановки диагноза в медицине, для выбора наиболее вероятной версии в юриспруденции и т. д.;
- *обучающие АИС* — всевозможные электронные учебники, компьютерные тесты, обучающие программы, а также тренажеры, имитирующие работу какого-либо устройства (самолета, автомобиля и т. д.).
- *системы автоматизированного проектирования* (САПР);
- *автоматизированные системы управления* (АСУ);
- ИС, обеспечивающие *автоматизацию документооборота и учета*.

Понятие «искусственный интеллект» было дано Джоном Маккарти в 1956 г. на конференции в Дартмутском университете.

Джон Маккарти (1927—2011) — американский информатик, автор термина «искусственный интеллект», изобретатель языка Лисп, основоположник функционального программирования, лауреат премии Тьюринга.

По словам Д. Маккарти: «Проблема состоит в том, что пока мы не можем в целом определить, какие вычислительные процедуры мы хотим называть интеллектуальными».

Деление автоматизированных информационных систем на виды достаточно условно, реальная АИС может сочетать в себе возможности систем разного вида. Большинство современных автоматизированных информационных систем в своей работе используют возможности компьютера и компьютерных сетей. Важными компонентами автоматизированных информационных систем являются базы и банки данных.

Важными направлениями развития современных информационных систем являются системы искусственного интеллекта (ИИ). **Искусственный интеллект** (ИИ; англ. artificial intelligence, AI) — свойство интеллектуальных систем выполнять творческие функции, которые традиционно считаются прерогативой человека. В этом направлении развивается робототехника, поисковые системы, системы автоматизированного управления и т. д. Результаты, полученные при создании и эксплуатации систем искусственного интеллекта, используются сейчас в системах распознавания речи и изображений, переводах текстов с одного языка на другой, постановке диагнозов, в сфере финансов и сфере компьютерных игр.

Существующие на сегодня интеллектуальные системы имеют очень узкие области применения. Например, программы, способные распознать лицо человека, не могут играть в шахматы. Развитие систем искусственного интеллекта, приближенного к человеческому, — задача будущего.

12.2. Информационные технологии

Определение информационных технологий тесно связано с понятием «технология». Термин **технология** происходит от греческого слова technē, означающего «наука об умении, мастерстве, искусстве». С помощью технологий описываются многие экономические, социальные, культурные и другие процессы, происходящие в обществе (пример 12.5).

Информационная технология — совокупность способов, приемов и методов сбора, обработки и передачи данных (первичная информация) для получения информации нового качества о состоянии объекта, явления или процесса (информационный продукт). Информационные технологии осуществляются во всех областях человеческой деятельности с использованием современных средств связи, вычислительной техники и программного обеспечения.

Инструментами информационных технологий являются аппаратное, программное и математическое обеспечение. С их помощью производится преобразование первичных данных в информационный продукт.

Технологический процесс по обработке информации обычно разбивают на этапы. Каждый этап состоит из операций и действий. В результате выполнения операций получают конкретный информационный продукт, который соответствует целям текущего этапа. Под действиями понимают совокупность приемов работы с конкретным программным обеспечением, приводящих к достижению поставленной цели (пример 12.6).

Пример 12.5. Сравнение технологии материального производства и информационной технологии:

Пример 12.6. Рассмотрим технологию создания стиля в текстовом документе.

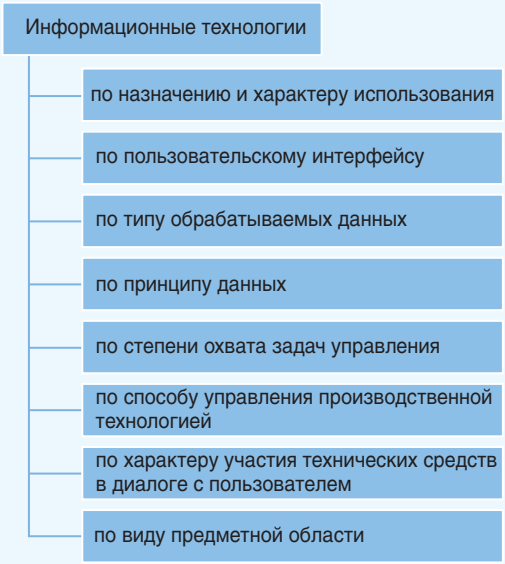
В качестве этапов можно выделить задание параметров абзаца, символа, применение текстовых эффектов и др.

Операции на этапе задания параметров символа — это выбор шрифта, цвета, размера, начертания символа.

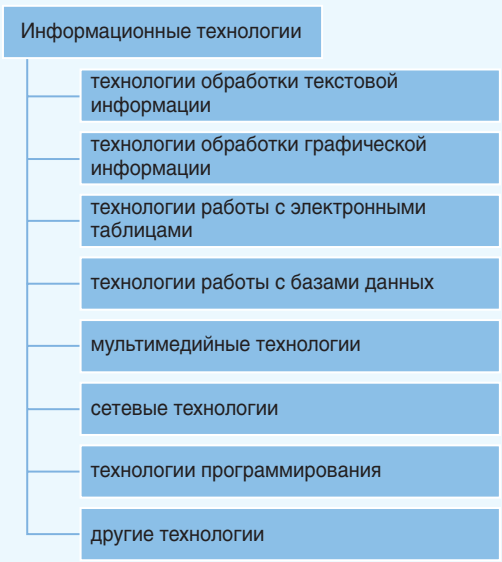
Действия — выбор определенных инструментов текстового редактора для производства операций: кнопки, списки, меню и др.

Преступления в сфере информационных технологий, или киберпреступность, — преступления, совершаемые людьми, использующими информационные технологии для преступных целей. Преступления в сфере информационных технологий включают как распространение вредоносных вирусов, взлом паролей, кражу номеров кредитных карточек и других банковских реквизитов (фишинг), так и распространение противоправной информации (клеветы, материалов порнографического характера, материалов, возбуждающих межнациональную и межрелигиозную вражду и т. д.) через Интернет.

Пример 12.7. Способы классификации информационных технологий по различным классификационным признакам:



Пример 12.8. Классификация информационных технологий по типу обрабатываемых данных:



Информационная технология, как и любая другая, должна:

- обеспечивать возможность разбиения всего процесса обработки на этапы, операции и действия;
- иметь регулярный характер, т. е. этапы, действия и операции технологического процесса должны быть стандартизированы, чтобы при каждом применении технологии получать однотипный результат, что позволит осуществлять целенаправленное управление информационными процессами.

Для грамотного использования информационных технологий в различных сферах жизни общества необходима их предварительная классификация. В зависимости от выбора классификационных признаков существуют различные классификации информационных технологий (пример 12.7).

В зависимости от выбора признака классификации информационные технологии могут делиться на разные группы (пример 12.8).

Для каждого вида информационных технологий определены один или несколько программных продуктов для конкретного типа компьютеров, технологии работы с которыми позволяют достичь поставленную пользователем цель по обработке данных (пример 12.9).

В настоящее время широкое развитие получили **облачные информационные технологии** — технологии распределенной обработки данных, в которой компьютерные ресурсы и

мощности предоставляются пользователю как интернет-сервис. Облачные технологии предоставляют пользователю возможности для хранения данных и программные средства для обработки данных. На сегодняшний день в облаке реализованы практически все информационные технологии, которые ранее использовались как локальные. Частью облачных технологий являются так называемые **грид-вычисления** (англ. *grid* — решетка, сеть) — это форма распределенных вычислений, в которой «виртуальный суперкомпьютер» представлен в виде кластеров, состоящих из отдельных компьютеров, соединенных с помощью сети, и работающих вместе для выполнения огромного количества заданий (операций, работ). Эта технология применяется для решения научных, математических задач, требующих значительных вычислительных ресурсов. Грид-вычисления используются также в коммерческой инфраструктуре для решения таких трудоемких задач, как экономическое прогнозирование, сейсмоанализ, разработка и изучение свойств новых лекарств и др.

Технологии грид-вычислений применяются сегодня и для обработки больших данных. Под **большими данными** (англ. *big data*) понимают структурированные и неструктурированные данные огромных объемов, эффективно обрабатываемые современными программными инструментами, которые появились в конце 2000-х гг. Определяющей характеристикой для больших данных

Пример 12.9. Связь информационных технологий с программными средствами обработки данных:

Термин «грид-вычисления» появился в начале 1990-х годов, как метафора, демонстрирующая возможность простого доступа к вычислительным ресурсам как к ресурсам электрической сети (англ. *power grid*). Использование свободного времени процессоров для решения исследовательских задач, требующих больших вычислительных мощностей и добровольного компьютинга (вычисление, выполняемое на компьютере), стало популярным в конце 1990-х гг. после запуска проектов добровольных вычислений GIMPS в 1996 г., distributed.net в 1997 г. и SETI@home в 1999 г. Идеи грид-системы (включая идеи из областей распределенных вычислений, объектно-ориентированного программирования, использования компьютерных кластеров, веб-сервисов и др.) были собраны и объединены Иэном Фостером, Карлом Кессельманом и Стивом Тики, которых часто называют отцами грид-технологии. Они начали создание набора инструментов для грид-компьютинга Globus Toolkit.

Пример 12.10. Характеристики больших данных:

- объем (англ. *volume*) характеризует величину физического объема данных;
- скорость (англ. *velocity*) определяет как стремительность прироста данных, так и необходимость высокоскоростной обработки данных для получения результатов;
- многообразие (англ. *variety*) позволяет одновременно обрабатывать различные типы структурированных и полуструктурированных данных.

Иногда характеристики расширяют («четыре V», «пять V» и т. д.). Кроме традиционных трех, еще добавляют: *veracity* — достоверность, *viability* — жизнеспособность, *value* — ценность, *variability* — переменчивость, *visualization* — визуализацию.

Пример 12.11. Устройства виртуальной реальности:

Очки виртуальной реальности

Шлем виртуальной реальности с сопутствующими аксессуарами

Комната виртуальной реальности

является набор признаков VVV («три V») — volume (объем), velocity (скорость), variety (многообразие) (пример 12.10). Технологии обработки больших данных являются альтернативой традиционным системам управления данными с помощью СУБД.

Неотъемлемой частью современного мира становятся технологии виртуальной и дополненной реальности.

Виртуальная реальность — созданный техническими средствами мир, передаваемый человеку через его ощущения: зрение, слух, обоняние, осязание и др. К устройствам виртуальной реальности относят шлемы, очки, комнаты виртуальной реальности и др. (пример 12.11).

Дополненная реальность (англ. *augmented reality*, AR — «расширенная реальность») — технологии, которые дополняют реальный мир, добавляя любые сенсорные данные. Дополненная реальность реализуется с помощью технологий компьютерного зрения, которые могут работать в «умных» очках и шлемах, мобильных устройствах, интерактивных стендах и др. (пример 12.12).

Технологии дополненной и виртуальной реальности используются в образовании и медицине, на их базе разрабатываются обучающие программы и тренажеры. Они нашли применение в программных продуктах для инженеров, архитекторов, дизайнеров, риелторов и др. Самый большой процент продаж программного обеспечения в

области виртуальной и дополненной реальности приходится на сферу развлечений: видеоигры, кино, анимация, сериалы и др.

Основываясь на научных прогнозах, можно говорить о том, что технологии виртуальной и дополненной реальности наряду с большими данными, облачными технологиями, искусственным интеллектом и некоторыми другими станут ключевыми технологиями очередной промышленной революции.

Большое количество информационных технологий разрабатывается в системах искусственного интеллекта (пример 12.13). **Машинное мышление** (англ. *machine reasoning*) охватывает процессы планирования, представления знаний и рассуждений, поиск и оптимизацию. Основой технологии **машинного обучения** (англ. *machine learning*, ML) является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. **Робототехника** включает в себя управление, восприятие, датчики и приводы, а также интеграцию всех методов в киберфизические системы. **Биологическое моделирование** искусственного интеллекта включает в себя такие направления, как нейронные сети, генетические алгоритмы, агентный подход.

Информационные технологии тесно связаны с информационными системами. Реализация функций информационной системы невозможна без применения ориентированной на

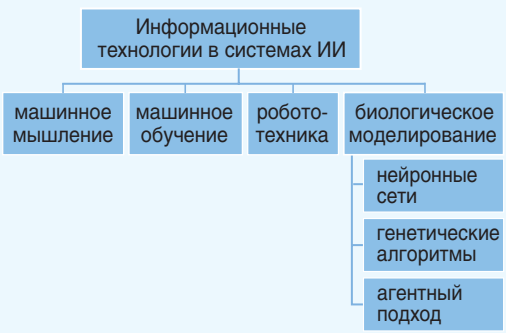
Пример 12.12. Устройства дополненной реальности:

Очки дополненной реальности

Дополненная реальность на мобильном устройстве

Стенды дополненной реальности

Пример 12.13. Информационные технологии в системах искусственного интеллекта:

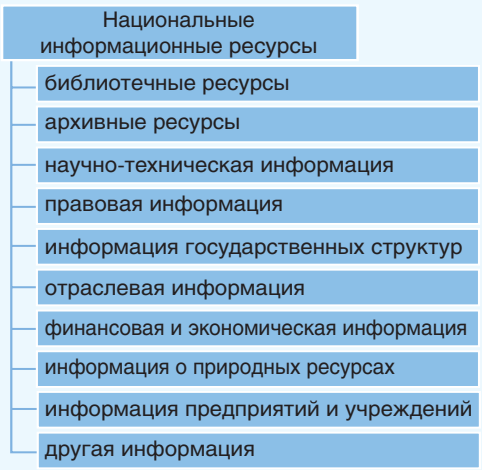


Нейронные сети используются для решения нечетких и сложных задач, таких как распознавание образов и классификация.

Генетический подход основан на идее, что некий алгоритм может стать более эффективным, если позаимствует лучшие характеристики у других алгоритмов («родителей»).

При агентном подходе создается программа — интеллектуальный агент — для взаимодействия с внешней средой. Примерами таких задач являются онлайн-торговля, ликвидация чрезвычайных ситуаций, моделирование социальных структур и др.

Пример 12.14. Национальные информационные ресурсы:



Пример 12.15. Информационные ресурсы общества обособлены от людей, которые их анализировали и создавали. Они материализовались в следующих видах:

- документы;
- книги;
- цифровые данные;
- алгоритмы;
- произведения искусства и науки.

нее информационной технологии, хотя сама технология может существовать и вне сферы информационной системы.

12.3. Информационные ресурсы

Традиционно в качестве ресурсов рассматриваются: материальные, природные, трудовые, финансовые, энергетические и др.

В современном обществе рассматривают информационные ресурсы — ресурсы, которые всегда существовали, но не рассматривались с точки зрения экономики.

Определение информационного ресурса дано в законе РБ «Об информации, информатизации и защите информации» (2008 г). **Информационный ресурс** — организованная совокупность документированной информации, включающая базы данных, другие совокупности взаимосвязанной информации в информационных системах (пример 12.14).

Документы не существуют сами по себе. В них в разной форме представлены знания, которыми обладали люди, создавшие эти документы. Поэтому информационными ресурсами являются знания, которые зафиксированы на материальном носителе и подготовлены людьми для социального использования (пример 12.15).

Материальные, финансовые и другие ресурсы измеряют и оценивают. В отношении информационных ресурсов критерии оценки их качества и количества пока не выработаны окончательно.

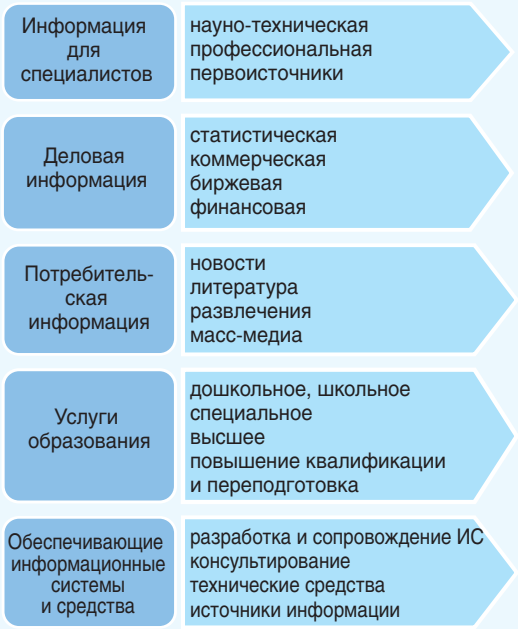
Развитие информационных ресурсов формирует рынок информационных услуг, позволяет превратить деятельность по оказанию информационных услуг в глобальную общечеловеческую деятельность (пример 12.16).

Информационные ресурсы являются основой для создания **информационных продуктов**, которые могут распространяться посредством **информационных услуг**. Информационные продукты и услуги формируют **информационный рынок** — систему экономических, правовых и организационных отношений по торговле продуктами интеллектуального труда.

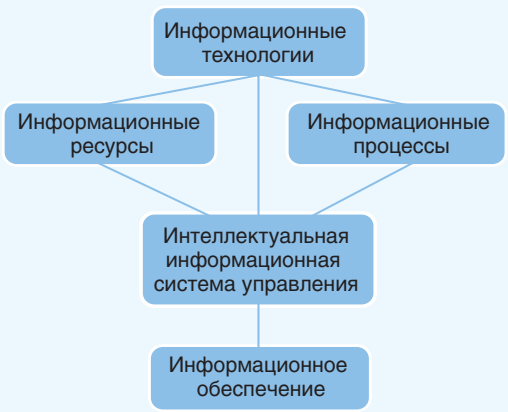
Основным источником информации для информационного обеспечения в современном обществе являются информационные системы, которые на основе баз данных интегрируют в себе поставщиков и потребителей информационных услуг. Информационные технологии, используя информационные ресурсы и процессы, обеспечивают порядок и условия предоставления информационных услуг, а также связи между ними (пример 12.17).

В настоящее время во всех развитых странах одним из главных направлений совершенствования системы государственного управления является переход на предоставление государственных услуг в электронном виде. В Беларуси создана и успешно развивается система предоставления таких услуг гражданам и бизнесу[1].

Пример 12.16. Рынок информационных услуг:



Пример 12.17. Взаимосвязь информационных технологий, информационных ресурсов, информационных процессов и информационного обеспечения:



[1] Национальный центр электронных услуг — https://nces.by/e-government/

1. Что такое информационная система?
2. Какие виды автоматизированных информационных систем вы можете назвать?
3. Что такое информационные технологии?
4. По каким признакам можно классифицировать информационные технологии?
5. Какие информационные технологии актуальны в современном мире?
6. Что такое информационные ресурсы?
7. Что такое информационные услуги?

Упражнения

1 Подготовьте сообщение на одну из тем:

1. Системы искусственного интеллекта.
2. Технологии виртуальной и дополненной реальности.
3. Технологии машинного обучения.
4. Рынок информационных услуг в Беларуси.

2 Найдите в Интернете значения следующих терминов: «высокие технологии», «биометрия», «ИТ-консалтинг», «метаданные», «чат-бот», «Data Science», «Интернет вещей», «умный город», «квантовая связь», «аутсорсинг и инсорсинг», «Мовазнаўчая беларусістыка», «информационное обслуживание», «блокчейн». Определите, к какому из понятий — «информационная система», «информационная технология», «информационный ресурс», информационная услуга — их можно отнести.

3 Ознакомьтесь с информационными ресурсами университетов Беларуси.

§ 13. Информатизация общества

Пример 13.1. Характеристики информационного общества:

- создание наукоемких производств и технологий с высокой интенсификацией компьютерного аппаратного, программного и информационного обеспечения;
- интенсивность информационных процессов;
- широкое использование телекоммуникаций;
- положительная динамика развития экономики;
- успехи в области экологической защиты человека.

Одна из целей развития информационного общества — создание широких возможностей для свободного развития личности и общества.

Основой **информационного общества** является широкое использование информационных технологий во всех сферах человеческой деятельности (пример 13.1).

Информационное общество отличается от традиционного индустриального общества высоким развитием информатики, связи и микроэлектронной промышленности, является источником новых рабочих мест, доминирующих в экономическом развитии.

Переход от индустриального общества к информационному сопровождается информатизацией.

Под **информатизацией** понимается организационный социально-экономи-

ческий и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей общества на основе формирования и использования информационных ресурсов.

Процесс информатизации базируется на достижениях таких наук, как кибернетика, информатика, микроэлектроника, теория систем и др.

В информационном обществе огромное значение приобретает формирование информационной культуры современного человека. Информационная культура является частью общечеловеческой культуры как совокупности материальных и духовных ценностей, созданных и создаваемых человечеством в процессе общественно-исторического развития.

Под **информационной культурой** понимается совокупность знаний и умений человека, представленных в виде системы правил его поведения в информационном обществе.

Современному человеку необходимо владеть теоретическими и практическими основами информатики, компьютерными устройствами, понимать и использовать информационные технологии в различных сферах человеческой деятельности: научной, производственной, социальной и др.

Процесс формирования информационной культуры человека включает в себя различные компоненты, такие как методологический, технологический, коммуникативный, языковый, алгоритмический и др. (пример 13.2).

Термин «информатизация» (англ. *informatization*) приобрел широкое распространение лишь в России и Китае. Это было связано, во-первых, с недостаточной разработанностью в 1980—1990-х гг. глоссария по тематике «информационные технологии» и «информационное общество», во-вторых, с некоторыми специфическими особенностями развития информационно-коммуникационных технологий в этих странах. Они характеризовались высоким уровнем развития прикладных, специализированных аппаратно-программных комплексов и крайне слабой телекоммуникационной инфраструктурой, которая становилась тормозом гармоничного развития информационного общества.

Пример 13.2. Компоненты информационной культуры.

Методологический компонент предполагает овладение человеком знаниями о роли и значении компьютерной техники, компьютерных информационных технологий и коммуникаций в жизни общества; понимание им экономических, социальных и психологических проблем, возникающих в информационном обществе и др.

Технологический компонент выражается в умении человеком использовать современные информационные технологии в своей деятельности.

Коммуникативный компонент включает в себя знания в области использования компьютерных сетей разного назначения, их технического и программного обеспечения, овладение человеком методами эффективного поиска информации в сети Интернет и др.

Языковый и **алгоритмический** компоненты предполагают знание и использование человеком одного либо нескольких языков программирования, основных понятий теории алгоритмов и т. д.

Пример 13.3. Основные черты информационной цивилизации[1].

1. Информационная экономика

Информационная сфера на сегодняшний день является одной из самых эффективных сфер вложения капитала. Общий объем мирового рынка информационных технологий оценивается величиной порядка 4 триллионов долларов[2] (по состоянию на 2020 г.), и этот объем постоянно растет.

2. Глобальная цифровизация

Одной из основных тенденций развития современной техники является широкое использование данных в цифровой форме. Обработка больших объемов данных и использование результатов их анализа позволят существенно повысить эффективность различных видов производства, технологий, оборудования, хранения, продажи, доставки товаров и услуг по сравнению с традиционными формами хозяйствования.

3. Развитие интеллектуальных компьютерных систем

Широкое распространение получают ряд компьютерных систем, экспертные системы с интеллектуальным интерфейсом, самообучающиеся системы. Возрастает роль интеллектуальных баз данных и машинного обучения.

4. Виртуализация экономики, политики и культуры

Появление рынка ценных бумаг привело к тому, что основные сделки стали совершаться не с реальными предметами, а с акциями, заменяющими эти предметы (т. е. по сути — с информационными моделями предметов). Замена реальных вещей их информационными моделями в информационной цивилизации становится всеобщим явлением.

Информационная культура имеет свое этическое и эстетическое содержание и выражается правилами, нормами поведения человека в информационном обществе, образцами деятельности, ценностными ориентирами и др.

Процесс построения информационного общества в каждой стране протекает по-разному. В разных странах принимаются различные законы о регулировании информационной деятельности человека, развитии индустрии информационных технологий (IT-индустрии).

В Беларуси IT-индустрия — самая динамично развивающаяся отрасль, которая с каждым годом превращается во все более мощный сектор экономики страны, сферу стратегического значения. В марте 2018 г. начал действовать подписанный Президентом Республики Беларусь документ, определяющий основные направления создания цифровой экономики XXI в. и развития Беларуси как IT-страны — декрет «О развитии цифровой экономики». Декрет открывает новые возможности для развития инноваций в Беларуси, достижений в самых передовых сферах от искусственного интеллекта до виртуальной реальности[3].

Процессы информатизации, проходящие в разных странах, в конечном итоге приведут к созданию информационной цивилизации (пример 13.3).

Информационная цивилизация — прямая «наследница» индустриальной цивилизации, которая сменила аграрную цивилизацию.

---

[1] Колин К. К. Информационная цивилизация. — Институт проблем информатики Ран Москва, 2002. — 112 с.

[2] http://www.tadviser.ru/index.php/Статья:ИТ_(мировой_рынок)

[3] http://president.gov.by/ru/official_documents_ru/view/dekret-8-ot-21-dekabrja-2017-g-17716/

Предполагается, что полноценная информационная цивилизация может быть только глобальной, охватывающей все человечество. Она должна базироваться на нравственной основе взаимного согласия, уступок и компромиссов при сохранении национально-культурных традиций, без навязывания чьих бы то ни было догм и мировоззрений, без войн и насилия.

Цифровизация и виртуализация общества ведет к некоторым негативным последствиям. В виртуальной политике важны не деловые качества кандидата, а его «имидж» (информационная модель). В искусстве таланта художника, писателя или исполнителя, как правило, оказывается недостаточно. Нужна значительная «раскрутка», требующая применения все тех же технологий.

1. Что такое информационное общество?
2. Какие черты информационного общества вы можете назвать?
3. Что такое информатизация?
4. Что понимают под информационной культурой?
5. Какие компоненты входят в информационную культуру?
6. Что такое информационная цивилизация?

Упражнения

Подготовьте сообщение на одну из тем:

1. Информационные революции в жизни общества.
2. Информатизация в Беларуси.
3. Перспективы развития информационной цивилизации.
4. Возможные опасности информационной цивилизации.

§ 14. Образование и профессиональная деятельность в информационном обществе

14.1. Образование в информационном обществе

В современном мире ключевое значение имеют знания. Для обозначения этого феномена используется термин «общество знаний» (пример 14.1).

В «обществе знаний» важнее всего «научиться учиться», а информационные технологии способствуют постоянному обновлению личной и профессиональной компетентности. Новые технологии повсеместно ускоряют создание и распространение знаний. Обучение

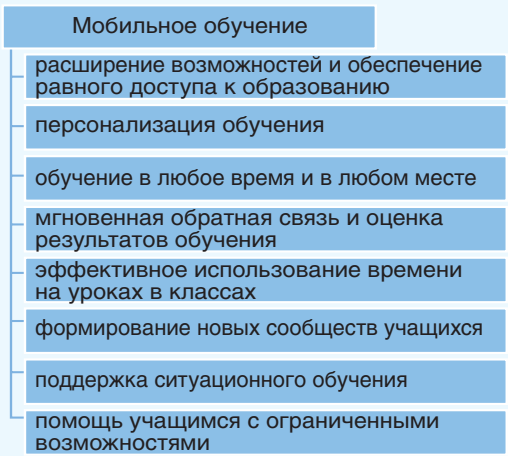
Пример 14.1. Характерные черты общества знаний:

- осознание роли знания как условия успеха в любой сфере деятельности;
- наличие постоянной потребности человека в новых знаниях, необходимых для решения новых задач, создания новых видов продукции и услуг;
- эффективное функционирование систем производства и передачи знаний;
- взаимное стимулирование предложения знаний и спроса на знания (предложение стремится удовлетворять имеющийся спрос на знания и формировать спрос).

Пример 14.2. Цели информатизации системы образования Республики Беларусь (согласно концепции информатизации):

- создание для населения равных возможностей получения качественных образовательных услуг на уровне современных требований национальных и международных стандартов вне зависимости от места проживания и обучения с использованием современных ИКТ;
- формирование личности, адаптированной к жизни в информационном обществе со всеми его возможностями, угрозами, вызовами и рисками.

Пример 14.3. Основные направления мобильного обучения:



Кроме традиционной очной и заочной формы обучения, некоторые белорусские университеты предлагают учиться дистанционно[1].

Дистанционная форма образования предполагает общение преподавателя и студента на расстоянии. Обмен информацией происходит с помощью Интернета, онлайн-сервисов. Учащийся может получить консультацию, необходимую литературу от преподавателя, послушать лекцию удаленно.

становится ключевой ценностью общества знаний. Согласно исследованиям ЮНЕСКО важное значение в обществе приобретают способность ориентироваться в потоке информации, когнитивные способности, критический ум, позволяющий отличать полезную информацию от бесполезной.

Для общества знаний важным является взаимосвязь знаний и системы образования, причем особую роль приобретают процессы информатизации в системе образования (пример 14.2).

Для системы образования актуальным становится лозунг: «Современный обучающийся — мобильный обучающийся!». Такой обучающийся: школьник, гимназист, лицеист, студент — должен иметь постоянный доступ к электронным образовательным ресурсам и услугам, в том числе в учреждении образования, дома, в дороге. Это касается и других участников образовательного процесса: родителей, педагогических работников, руководителей системы образования разных уровней. Мобильность каждого участника образовательного процесса будет лежать в основе мобильного образования в новом информационном обществе (пример 14.3).

Обеспечить актуальность и мобильность образовательных ресурсов позволит применение «облачных» технологий, разработка и внедрение электронных образовательных ресурсов.

Одной из форм электронного обучения является дистанционное обучение. Оно предоставляет возможность учиться вне зависимости от места работы

[1] Адукар. https://adukar.by/news/distancionnoe-obuchenie

и проживания, для него характерны гибкость и экономичность. К перспективным направлениям дистанционного обучения можно отнести дополнительное образование взрослых (повышение квалификации и переподготовку, обучающие курсы, подготовку к поступлению в учреждения образования и др.), дополнительное образование детей и молодежи, специальное образование.

Контроль знаний также проводится дистанционно: по видеосвязи или с помощью интерактивных программ тестирования. Однако сессия сдается в вузе.

Главные преимущества: учеба в любое время, в удобном месте и психологически комфортной обстановке. Студент сам выбирает приемлемый темп занятий, благодаря чему легко совмещать учебу с другими делами.

14.2. Профессиональная деятельность в информационном обществе

В информационном обществе главным ресурсом является информация. Именно на основе владения информацией о самых различных процессах и явлениях можно эффективно и оптимально строить любую деятельность. При этом повышается не только качество потребления, но и качество производства; человек, использующий информационные технологии, имеет лучшие условия труда, труд становится интеллектуальным, творческим, креативным.

Основным критерием развитости информационного общества можно считать количество населения, занятого в информационной сфере, а также использующего информационные и коммуникационные технологии в своей повседневной деятельности.

Все больше жителей Беларуси используют Интернет в профессиональной сфере и в повседневной жизни. По подсчетам аналитиков, к январю

По данным Национального статистического комитета Республики Беларусь, доля населения, занятого в ИТ-сфере, увеличилась с 1,7 % (2010 г.) до 2,5 % (2018 г.). При этом в 2019 г. доля IT-сектора в ВВП составляет 5,5% и по прогнозам может вырасти до 10 % к 2023 г.

Пример 14.4. Обзор перспективных профессий:

• Прогноз от Адукара[1]. Востребованными останутся специалисты по информационной безопасности, оценке интеллектуальной собственности, рекламщики, маркетологи, логисты. В будущем понадобятся специалисты в области искусственного интеллекта, биоинженеры, урбанисты, строители умных городов и дорог.

• Востребованные профессии 2020[2]. Среди общего списка наиболее нужных профессий 2015—2020 гг. называют: IT-специалистов (программисты, специалисты по администрированию баз данных или серверного оборудования, веб-дизайнеры), инженеров и специалистов в сфере маркетинга, медиков, педагогов, переводчиков.

[1] https://adukar.by/news/kakie-professii-budut-vostrebovany-v-budushchem

[2] http://www.proprof.ru/stati/careera/vybor-professii/statistika-i-reytingi/vostrebovannye-professii-2020-spisok

Пример 14.4. *Продолжение.*

• Самые востребованные профессии будущего десятилетия[1]. Такими профессиями стали IT-специалисты, инженеры, маркетологи, медики, специалисты в области сервиса, специалисты в области нанотехнологий.

Одной из самых необычных профессий будущего называют специалиста по хранению памяти. Предполагается, что ведущие нейробиологи пустят в массы революционные мозговые импланты, которые позволят человеку использовать мозг как флэшку. Специалисты будут помогать людям рационально работать со своей памятью, не перегружая мозги[2].

В информационном обществе следует ожидать востребованности профессий информационной ориентации:

• инженеры знаний — специалисты, свободно ориентирующиеся в автоматизированных системах формирования, хранения и использования информационных ресурсов;

• системные аналитики, обладающие опытом компьютерного моделирования и прогнозирования в различных областях социальной практики;

• информационные менеджеры, предлагающие свои услуги в обеспечении информацией различных направлений деятельности.

2019 г. на 9,44 млн жителей Беларуси приходилось 11,87 млн абонентов мобильной связи — это примерно 126 % населения. Показатель увеличился на 3,1 % (357 тыс.) по сравнению со статистикой на январь 2018 г.[3].

В настоящее время достаточно много интернет-ресурсов предлагают обзоры профессий, которые будут актуальными в ближайшее десятилетие (пример 14.4).

Во всех списках на первом месте — специалисты IT-сферы. К профессиям этой сферы относятся: программисты, специалисты по администрированию баз данных или серверного оборудования, веб-дизайнеры, специалисты по информационной безопасности, специалисты по оценке интеллектуальной собственности и др.

В списках также фигурируют специальности инженера, медика, химика, логиста, нанотехнолога — те специальности, которые требуют прочных знаний в области естественных наук и математики.

1. Какие характерные черты общества знаний вы можете назвать?
2. Что понимают под мобильным обучением?
3. В чем преимущества дистанционного обучения?
4. Что считают основным критерием развитости информационного общества?
5. Какие профессии будут востребованы в ближайшем будущем?

Упражнения

Подготовьте сообщение на одну из тем:

1. Дистанционное обучение в белорусских вузах.
2. Необычные профессии будущего.

[1] https://benefit.by/page/show/articles/2285

[2] https://gagadget.com/science/23845-10-neobyichnyih-professij-buduschego-kotoryie-poyavyatsya-uzhe-zavtra/

[3] https://dev.by/news/digital-2019-belarus

§ 15. Кибербезопасность и киберустойчивость

15.1. Кибербезопасность

Термин «кибербезопасность» объединяет в себе слова «кибер» и «безопасность». Начнем с безопасности.

Безопасность — это состояние защищенности сбалансированных интересов личности, общества и государства от внешних и внутренних угроз.

Угроза безопасности — это возможность воздействия на систему или объект, которое может нарушить безопасность.

В зависимости от сфер интересов различают несколько видов безопасности. В сфере экономики — экономическая безопасность, в сфере экологии — экологическая безопасность и т. д.

Ключевое значение в современном мире приобрела информационная сфера, которая оказывает всеобъемлющее влияние на проходящие экономические, политические и социальные процессы.

Информационная безопасность — это состояние защищенности сбалансированных интересов личности, общества и государства от внешних и внутренних угроз в информационной сфере.

Понятие информационной безопасности введено у нас в стране на законодательном уровне (пример 15.1).

Основными объектами информационной сферы являются *информационные системы*, которые включают банки данных, информационные технологии и комплекс программно-технических средств (пример 15.2).

Пример 15.1. Термин «информационная безопасность» введен в Концепции информационной безопасности Республики Беларусь, утвержденной Советом Безопасности Республики Беларусь в 2019 году.

В Концепции целью обеспечения информационной безопасности поставлено достижение и поддержание такого уровня защищенности информационной сферы, который обеспечивает реализацию национальных интересов Республики Беларусь и ее прогрессивное развитие.

Внимание к информационной безопасности на государственном уровне вызвано тем, что объекты промышленности, транспорта, энергетики, электросвязи, здравоохранения построены на базе компьютерных систем. Это ставит жизнь и здоровье населения в прямую зависимость от надежности этих компьютерных систем и защищенности информации в них.

Пример 15.2. Термин «информационная система» введен в Законе Республики Беларусь «Об информации, информатизации и защите информации», принятом в 2008 году.

Информационные системы предназначены для хранения, обработки, поиска, распространения, передачи и представления информации. По степени распределенности они делятся на локальные (настольные) и распределенные.

Примеры локальных информационных систем — это пакеты офисных программ (MS Office, OpenOffice и др.), системы программирования, системы управления базами данных (СУБД).

В распределенных информационных системах компоненты распределены по нескольким компьютерам. Примерами могут быть информационные системы предприятий, учреждений и организаций, автоматизированные системы управления производственными процессами, базы данных и информационно-поисковые системы в Интернете.

Пример 15.3. Слово «кибер» возникло в английском языке как часть слова «кибернетика», которое является названием науки об общих законах управления, связи и переработки информации.

Слово «кибер» вначале обозначало слово «кибернетический» и впервые стало использоваться писателями-фантастами. Именно в фантастике возникло слово «киберпространство».

В русскоязычной фантастике киберами называли разновидность роботов.

Пример 15.4. Киберпространство — понятие для обозначения виртуальной реальности.

Киберспорт (компьютерный спорт) — командные или индивидуальные соревнования по компьютерным видеоиграм, которые имитируют реальные спортивные состязания.

Кибербуллинг — это анонимные намеренные оскорбления, угрозы, нападки на человека при помощи Интернета.

Кибертерроризм — это воздействия на информационные системы, несущие угрозу здоровью и жизни людей, применяемые для устрашения населения либо для дестабилизации общественного порядка.

Пример 15.5. *Целостность информации* означает, что изменение информации может быть только контролируемым.

Конфиденциальность информации означает, что лицо, получившее к ней доступ, не имеет права передавать ее другим лицам.

Подлинность информации означает, что точно установлено авторство информации.

Доступность информации означает, что лицо, имеющее доступ к информации, реализует этот доступ беспрепятственно.

Сохранность информации означает, что при любых обстоятельствах информация не будет безвозвратно потеряна.

Технические средства, системы и технологии хранения, передачи и обработки информации образуют *информационную инфраструктуру* предприятия, организации или государства.

Кибербезопасность — это состояние защищенности информационной инфраструктуры и содержащейся в ней информации от внешних и внутренних угроз.

Таким образом, кибербезопасность — это раздел информационной безопасности.

«Кибер» — приставка, которая означает «связанный с компьютерами и Интернетом» и имеет интересную историю (пример 15.3). Известно немало терминов, которые используют это слово-приставку (пример 15.4).

Кибербезопасность предполагает защищенность информации в информационной инфраструктуре. В соответствии с Законом Республики Беларусь «Об информации, информатизации и защите информации» защита информации должна обеспечивать *целостность*, *конфиденциальность*, *подлинность*, *доступность* и *сохранность* информации. В примере 15.5 приведены описания этих ее свойств.

Защита информации — это не простая задача, так как количество данных со временем растет лавинообразно. Только с 2010 по 2020 год объем хранимой во всем мире информации вырос в 50 раз. Число серверов американских компаний Google и Amazon исчисляется миллионами.

15.2. Уязвимость информационной инфраструктуры

Кибербезопасность предполагает защищенность информационной инфраструктуры и содержащейся в ней информации от внешних и внутренних угроз.

Угрозы возникают, если информационная инфраструктура имеет уязвимость.

Уязвимость — свойство информационной инфраструктуры или ее объектов, которое позволяет реализовать угрозу (воздействовать на инфраструктуру).

Воздействие на информационную инфраструктуру или ее объекты может быть *случайным* или *умышленным* (пример 15.6).

Умышленное воздействие на информационную инфраструктуру программными или программно-техническими средствами называется ***кибератакой***.

Основная цель умышленных воздействий — получение финансовой выгоды. Для этого используют как вымогательство, так и хищение ценной информации (персональные данные аккаунтов, данные доступа к платежным сервисам).

Уязвимость могут иметь информационная инфраструктура в целом, ее технические и программные средства, персонал и пользователи.

Если антивирусные средства инфраструктуры имеют уязвимость, то вирусная кибератака может достигнуть успеха (пример 15.7).

Пример 15.6. Случайное воздействие может являться следствием ошибочных действий персонала, внезапных отказов технических средств и т. д.

Умышленные воздействия подразделяются на активные и пассивные.

Пассивные воздействия направлены на несанкционированное использование информационных ресурсов, например подслушивание, видеозапись.

Активными умышленными воздействиями являются внедрение вредоносных программ в компьютеры пользователей, мошенническое использование сайтов информационных услуг и др.

Простейший и распространенный способ умышленного воздействия — это фишинг (упоминался в пособии для 9-го класса). Представляет собой рассылку электронных писем или сообщений в соцсетях от имени коллег по работе, сотрудников банка или представителей государственного учреждения. Цель фишингового письма — заставить получателя немедленно перейти на поддельный веб-сайт, весьма похожий на известный, и там ввести конфиденциальную информацию, чтобы избежать якобы внезапно возникшую опасность или каких-то серьезных последствий бездействия. Умышленные воздействия, реализующие угрозы кибербезопасности, в соответствии с Уголовным кодексом Республики Беларусь могут быть квалифицированы как преступления против информационной безопасности.

Пример 15.7. Вирусы заражают файлы специфическим кодом. Троянцы — вредоносные программы, которые прячутся под маской легального программного средства. Шпионские программы тайно следят за действиями пользователя и собирают финансовую персональную информацию. Программы-вымогатели шифруют файлы и данные, а преступники требуют выкуп за восстановление.

Пример 15.8. Уязвимость программного средства может быть результатом ошибок программирования, но чаще это непредусмотренное свойство, которое разработчику выявить очень сложно. Проверкой работоспособности программ занимаются тестировщики, а поиском уязвимости — злоумышленники.

Пример 15.9. В приложениях пакета MS Office уязвимость находят постоянно, и разработчики постоянно создают и рассылают обновления, которые закрывают найденную уязвимость.

Это противостояние злоумышленников и разработчиков ведется уже давно.

С появлением новых информационных технологий и новых программных средств можно ожидать появления новых уязвимостей и новых угроз.

Пример 15.10. В августе 2013 г. были взломаны аккаунты с персональными данными более 3 млрд пользователей компании Yahoo!.

В 2018 г. оказались взломанными аккаунты более 500 млн клиентов гостиничной сети Marriott, 440 млн пользователей программного обеспечения Veeam, 300 млн клиентов логистической компании SF Express.

В августе 2020 года в Интернете появилась незащищенная база данных, содержащая персональные данные около 235 млн пользователей сервисов Instagram, TikTok и YouTube.

Пример 15.11. Успех распределенных сетевых атак основан на уязвимости серверов компьютерной сети, которые всегда имеют ограничения по количеству одновременно обрабатываемых запросов. Если число запросов превышает предельные возможности сервера, то результатом может быть полное прекращение его нормальной работы — отказ в обслуживании.

Большой проблемой являются уязвимость операционных систем и приложений (пример 15.8). Как только злоумышленники находят слабое место программы, они создают свой зловредный программный код, который называется «эксплойт», и, атакуя, пытаются внедрить его в программу (пример 15.9).

По оценкам экспертов, кибератакам эксплойтов-вымогателей компании во всем мире подвергаются каждые 14 с. Масштабы хищения персональных данных пользователей с серверов сети впечатляют еще больше (пример 15.10).

Другой вид кибератак — *распределенные сетевые атаки*. Их называют также атаками типа «отказ в обслуживании» (по-английски Distributed Denial of Service) или DDoS-атаками (пример 15.11).

Обычная цель DDoS-атаки — вымогательство денег за прекращение атаки, дискредитация бизнес-конкурента или нанесение ему ущерба.

С каждым годом число кибератак растет. При этом набор приемов, которые используют злоумышленники, пополняется почти ежемесячно.

Но самым уязвимым местом кибербезопасности был и будет человеческий фактор — поведение сотрудников и пользователей.

Согласно статистике 91 % атак на банки совершают коррумпированные сотрудники самих банков, 8 % — банковские посредники и только 1 % приходится на хакеров.

По результатам исследований, потери российского пользователя в ходе ки-

бератак составляют в среднем 80 долларов, а каждая девятая жертва теряет более 1000 долларов (пример 15.12).

Для получения конфиденциальной информации злоумышленники широко используют социальную инженерию. Это психологическое манипулирование людьми с целью совершения определенных действий или разглашения конфиденциальной информации (пример 15.13).

15.3. Обеспечение кибербезопасности. Киберустойчивость

Грамотный подход к кибербезопасности предполагает реализацию системы мер защиты информационной инфраструктуры.

Обеспечение кибербезопасности — это система мер противодействия угрозам информационной безопасности.

Следует заметить, что в Интернете понятие «обеспечение кибербезопасности» часто называют кибербезопасностью (в широком смысле).

Обеспечение кибербезопасности — это воплощение всех мер защиты сетей, устройств и приложений, защиты конфиденциальности и целостности данных, а также мер сохранения нормальной работы информационной инфраструктуры в целом (пример 15.14).

В системе мер обеспечения кибербезопасности особое место занимает обучение пользователей. Пользователь и его цифровое устройство — это популярная цель злоумышленников. Взломав индивидуальное цифровое

Пример 15.12. Причиной потери 21 % пострадавших назвали предоставление мошенникам доступа к записям в платежном сервисе, 19 % — ввод своих конфиденциальных данных на поддельном веб-сайте.

Пример 15.13. Мошенник вступает в контакт с пользователем, притворяясь сотрудником банка, продавцом или покупателем Интернет-сервиса, доверенным лицом сервиса и т. д. Цель — психологически вынудить пользователя сообщить свои персональные данные злоумышленнику или на сайт, который злоумышленник указывает.

Пример 15.14. Система мер противодействия угрозам информационной безопасности образует несколько уровней защиты как внутри инфраструктуры, так и на ее периметре. На каждом уровне реализуются определенные принципы и средства контроля.

Первую линию защиты сетей обеспечивают межсетевые экраны, которые осуществляют мониторинг входящего и исходящего сетевого трафика. Межсетевой экран может быть аппаратным, программным или смешанного типа.

Стратегия DLP (data-loss prevention), которую реализуют специалисты по информационной безопасности, позволяет контролировать возможные пути утечки данных.

Защиту информации обеспечивают системы идентификации по биометрическим данным, системы криптографической защиты каналов передачи и носителей данных, программные решения для управления ключами шифрования.

В последние годы шифрование данных стало нормой в работе с веб-сайтами.

Пример 15.15. В учебном пособии «Информатика» для 9-го класса был приведен ряд эффективных мер безопасности пользователя в сети Интернет. В дополнение к ним приведем следующие меры:

1. Регулярно выполняйте резервное копирование данных, которое дает гарантии сохранности уникальной информации. В случае кражи или шифрования такой информации вымогательство не пройдет. Восстановить информацию из резервных копий можно в любой момент.

2. Обновляйте операционную систему приложения. Используя обновления, вы получаете свежие исправления уязвимости.

3. Внимательно относитесь к тому, что подключаете к компьютеру. Вредоносные программы могут распространяться через зараженные флешки, внешние жесткие диски и даже смартфоны.

4. Избегайте незащищенных сетей Wi-Fi в общественных местах. Подключившись к незащищенной сети Wi-Fi, вы можете подвергнуться атаке Man-in-the-Middle («Человек посередине»). Это атака, в ходе которой злоумышленник перехватывает данные во время их передачи. Он как бы становится промежуточным звеном в информационной цепочке, а жертва об этом даже не подозревает и может передать ценную информацию.

5. Банковские операции или покупки выполняйте только на принадлежащем вам устройстве в сети, которой вы доверяете.

6. Следите за сохранностью своих цифровых устройств (смартфона, планшета и ноутбука). Не допускайте к их использованию незнакомцев.

7. Как правило, стереть данные из Интернета невозможно. Все, что попало в Интернет, останется там навсегда. Единственный способ избежать утечки информации — не делиться ею.

устройство, можно получить доступ не только к информации в нем, но и к конфиденциальной информации в сети.

Меры обеспечения кибербезопасности для пользователей часто формулируют как меры или правила безопасности пользователя (пример 15.15).

Эффективная система мер противодействия угрозам информационной безопасности переводит информационную инфраструктуру в новое состояние — в состояние киберустойчивости.

Киберустойчивость — способность информационной инфраструктуры успешно предотвращать реализацию угроз или быстро восстанавливаться после их реализации.

Кибератака на уровне пользователя может привести к самым разным последствиям.

Технические решения не смогут защитить от угроз, если злоумышленник-профессионал завладеет вашим доверием. Многие из них действительно изучают поведение людей, собирают информацию, изучают методы, которыми люди пользуются, и на основе этих данных совершают киберпреступления. Поэтому главное — не доверяйте незнакомцам и соблюдайте бдительность.

Поймите, что вы — привлекательная цель для злоумышленников, и кибератака может случиться в любое время, в любом месте, на любом устройстве. Вам поможет только ваша готовность отразить эту атаку.

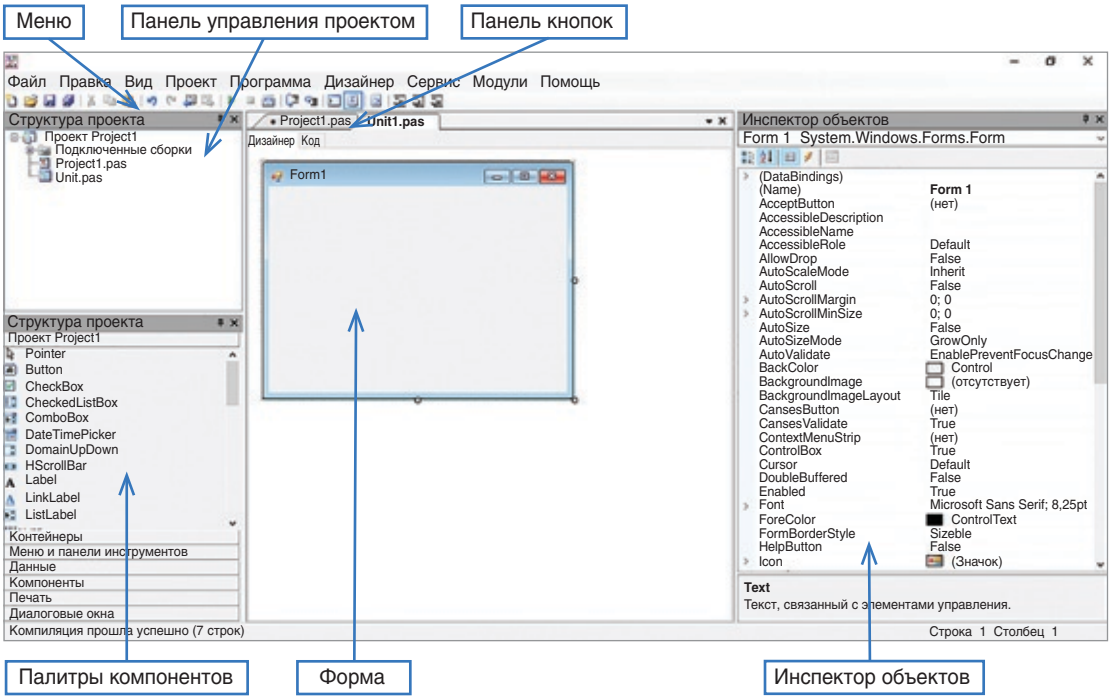
1. Что такое безопасность?
2. Что такое угроза безопасности?
3. Что такое информационная безопасность?
4. В каком законодательном документе введено понятие «информационная система»?
5. Что такое кибербезопасность?
6. Что означает слово-приставка «кибер»?
7. Как произносится слово «cyber» в английском языке?
8. Почему защита информации в современном мире является сложной задачей?
9. Что такое уязвимость информационной инфраструктуры?
10. Что такое кибератака?
11. Что скрывается за аббревиатурой DDoS?
12. Что такое социальная инженерия?
13. В чем схожи фишинг и социальная инженерия?
14. Что такое обеспечение кибербезопасности?
15. Что такое киберустойчивость?
16. Почему обучение пользователей занимает особое место в системе мер обеспечения кибербезопасности?
17. К каким последствиям может привести кибератака на уровне пользователя?

Упражнения

1. Приведите примеры умышленных воздействий на информационные инфраструктуры.
2. Приведите примеры кибератак.
3. Приведите другие названия DDoS-атак.
4. Назовите возможные цели DDoS-атак.
5. Найдите в Интернете определение эксплойта.
6. Найдите в Интернете и приведите примеры видов эксплойтов.
7. Выясните в Интернете, что такое претекстинг.
8. Выясните в Интернете определение и происхождение слова «хакер».

ПРИЛОЖЕНИЕ К ГЛАВЕ 1

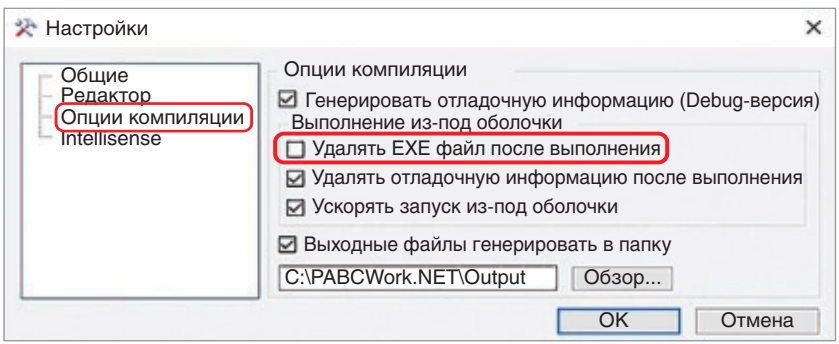
Окно среды PascalABC.Net (при создании проекта Windows Form)



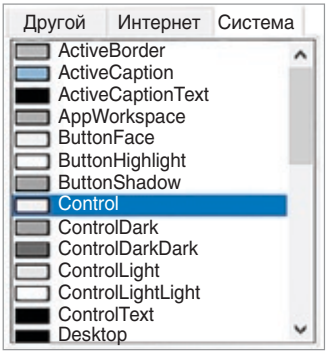
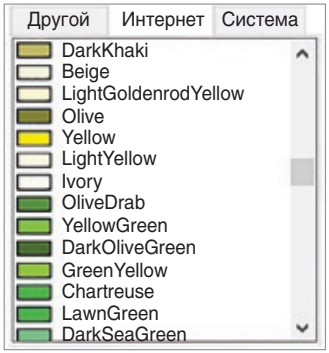
Опции компилятора, при которых сохраняется exe файл

Открыть окно **Настройки** (команда меню **Сервис** → **Настройки**).

В разделе **Опции компиляции** снять галочку с опции **Удалять EXE файл после выполнения.**



Цветовые константы

Система	Интернет	Палитра
		

Работа с графикой

Свойство **Pen** (карандаш) позволяет определять цвет, толщину линий. Создание:

`Pen p_c := new Pen(Color.Black, 1);` — карандаш черного цвета, толщиной в 1 пиксель.

Свойство **Brush** (кисть) предназначено для заполнения областей заданным шаблоном. Создание: `SolidBrush sb := new SolidBrush(Color.Blue);` — кисть синего цвета.

Методы Graphics позволяют строить графические изображения. Во многом методы похожи на аналогичные команды построения графических изображений в среде PascalABC.Net.

Краткое описание некоторых методов graphics	
`DrawArc(Pen, Rectangle, Single, Single)`	Рисует дугу, которая является частью эллипса, заданного структурой Rectangle и двумя радиальными линиями
`DrawEllipse(Pen, Rectangle)`	Рисует эллипс
`DrawLine(Pen, Point, Point)`	Проводит линию, соединяющую две структуры Point

Краткое описание некоторых методов graphics	
`DrawPie(Pen, Rectangle, Single, Single)`	Рисует сектор, который определяется эллипсом, заданным структурой Rectangle и двумя радиальными линиями
`DrawRectangle(Pen, Rectangle)`	Рисует прямоугольник, определяемый структурой Rectangle
`DrawString(String, Font, Brush, PointF)`	Создает указываемую текстовую строку в заданном месте с помощью определяемых объектов Brush и Font
`FillEllipse(Brush, Rectangle)`	Заполняет внутреннюю часть эллипса, определяемого ограничивающим прямоугольником, который задан структурой Rectangle
`FillPie(Brush, Rectangle, Single, Single)`	Заполняет внутреннюю часть сектора, определяемого эллипсом, который задан структурой Rectangle, и двумя радиальными линиями
`FillPolygon(Brush, Point[])`	Заполняет внутреннюю часть многоугольника, определяемого массивом точек, заданных структурами Point
`FillRectangle(Brush, Rectangle)`	Заполняет внутреннюю часть прямоугольника, определяемого структурой Rectangle

Структура **Rectangle** описывается двумя парами координат, являющихся диагональю прямоугольника со сторонами, параллельными границам экрана.

Методы **DrawArc**, **DrawPie** и **FillPie** имеют одинаковые параметры. Радиальные линии задают угол (в градусах), который отмеряется по часовой стрелке, начиная от оси X и заканчивая точкой дуги.

Обработчики событий приложения «Графический редактор»

```
procedure Form1.Form1_Load(sender: Object; e: EventArgs);
begin
 bm := new Bitmap(pictureBox1.Width, pictureBox1.Height);
 pictureBox1.Image := (Image)(bm);
 gr := Graphics.FromImage(pictureBox1.Image);
 gr.Clear(Color.White);
 gr := pictureBox1.CreateGraphics;
 c_f := Color.Black;
 p_c := new Pen(c_f, 1);
 c_b := Color.White;
 s_b := new SolidBrush(c_b);
end;

procedure Form1.numericUpDown1_ValueChanged(sender: Object;
e: EventArgs);
begin
 p_c.Dispose;
 w := numericUpDown1.Value;
 p_c := new Pen(c_f, (integer)(w));
end;

procedure Form1.button1_Click(sender: Object; e: EventArgs);

begin
 colorDialog1.ShowDialog();
 c_f := colorDialog1.color;
 w := numericUpDown1.Value;
 p_c.Dispose;
 p_c := new Pen(c_f, (integer)(w));
 panel1.BackColor := c_f;
end;

procedure Form1.button2_Click(sender: Object; e: EventArgs);
begin
 colorDialog1.ShowDialog();
 c_b := colorDialog1.color;
 s_b.Dispose;
 s_b := new SolidBrush(c_b);
 panel2.BackColor := c_b;
end;

procedure Form1.pictureBox1_MouseDown(sender: Object;
e: MouseEventArgs);
begin
 m_d := true;
 x1 := e.x; y1 := e.y;
end;

procedure Form1.pictureBox1_MouseMove(sender: Object;
e: MouseEventArgs);
begin
 if m_d then
```

```
 begin
  gr.DrawLine(p_c, x1, y1, e.X, e.Y);
 end;
 //запомнить координаты для рисования следующего отрезка
 x1 := e.X; y1 := e.Y
end;

procedure Form1.pictureBox1_MouseUp(sender: Object;
e: MouseEventArgs);
begin
 m_d := false;
end;

procedure Form1.toolStripMenuItem3_Click(sender: Object;
e: EventArgs);
begin
 gr := pictureBox1.CreateGraphics;
 gr.Clear(c_b);
end;

procedure Form1.toolStripMenuItem4_Click(sender: Object;
e: EventArgs);
begin
 openFileDialog1.ShowDialog();
 F_N := openFileDialog1.FileName;
 Text := 'myPaint ' + F_N;
 PictureBox1.Load(F_N);
end;

procedure Form1.toolStripMenuItem5_Click(sender: Object;
e: EventArgs);
begin
 saveFileDialog1.ShowDialog();
 F_N := saveFileDialog1.FileName;
 PictureBox1.Image.save(F_N);
 Text := 'myPaint ' + F_N;
end;

procedure Form1.numericUpDown1_ValueChanged (sender: Object;
e: EventArgs);
begin
 p_c.Dispose;
 w := numericUpDown1.Value;
 p_c := new Pen(c_f, (integer)(w));
end;

procedure Form1.toolStripMenuItem7_Click(sender: Object;
e: EventArgs);
begin
 close;
end;
```

Кодирование цвета

Название цвета	Образец	Код
black		#000000
dimgray		#696969
gray		#808080
darkgray		#a9a9a9
silver		#c0c0c0
lightgrey		#d3d3d3
whitesmoke		#f5f5f5
white		#ffffff
rosybrown		#bc8f8f
lightcoral		#f08080
brown		#a52a2a
firebrick		#b22222
maroon		#800000
darkred		#8b0000
red		#ff0000
salmon		#fa8072
mistyrose		#ffe4e1
tomato		#ff6347
darksalmon		#e9967a
coral		#ff7f50

Название цвета	Образец	Код
orangered		#ff4500
lightsalmon		#ffa07a
sienna		#a0522d
seashell		#fff5ee
saddlebrown		#8b4513
chocolate		#d2691e
peachpuff		#ffdab9
sandybrown		#f4a460
linen		#faf0e6
bisque		#ffe4c4
darkorange		#ff8c00
antiquewhite		#faebd7
tan		#d2b48c
burlywood		#deb887
blanchedalmond		#ffebcd
papayawhip		#ffefd5
oldlace		#fdf5e6
wheat		#f5deb3
orange		#ffa500
floralwhite		#fffaf0

Название цвета	Образец	Код
goldenrod		#daa520
darkgoldenrod		#b8860b
cornsilk		#fff8dc
gold		#ffd700
lemonchiffon		#fffacd
khaki		#f0e68c
palegoldenrod		#eee8aa
darkkhaki		#bdb76b
ivory		#fffff0
beige		#f5f5dc
lightyellow		#ffffe0
olive		#808000
yellow		#ffff00
olivedrab		#6b8e23
yellowgreen		#9acd32
darkolivegreen		#556b2f
greenyellow		#adff2f
lawngreen		#7cfc00
chartreuse		#7fff00
honeydew		#f0fff0
darkseagreen		#8fbc8f
lightgreen		#90ee90
palegreen		#98fb98

Название цвета	Образец	Код
forestgreen		#228b22
limegreen		#32cd32
darkgreen		#006400
green		#008000
lime		#00ff00
seagreen		#2e8b57
mediumseagreen		#3cb371
mintcream		#f5fffa
springgreen		#00ff7f
mediumspringgreen		#00fa9a
mediumaquamarine		#66cdaa
aquamarine		#7fffd4
turquoise		#40e0d0
lightseagreen		#20b2aa
mediumturquoise		#48d1cc
azure		#f0ffff
paleturquoise		#afeeee
darkslategray		#2f4f4f
teal		#008080
darkcyan		#008b8b
cyan		#00ffff
lightcyan		#e0ffff
darkturquoise		#00ced1

Название цвета	Образец	Код
powderblue		#b0e0e6
lightblue		#add8e6
deepskyblue		#00bfff
skyblue		#87ceeb
lightskyblue		#87cefa
steelblue		#4682b4
aliceblue		#f0f8ff
slategray		#708090
lightslategray		#778899
dodgerblue		#1e90ff
lightsteelblue		#b0c4de
cornflowerblue		#6495ed
royalblue		#4169e0
ghostwhite		#f8f8ff
lavender		#e6e6fa
midnightblue		#191970
navy		#000080
darkblue		#00008b
mediumblue		#0000cd
blue		#0000ff
darkslateblue		#483d8b
slateblue		#6a5acd

Название цвета	Образец	Код
mediumslateblue		#7b68ee
mediumpurple		#9370db
blueviolet		#8a2be2
indigo		#4b0082
darkorchid		#9932cc
darkviolet		#9400d3
mediumorchid		#ba55d3
thistle		#d8bfd8
plum		#dda0dd
violet		#ee82ee
purple		#800080
darkmagenta		#8b008b
magenta		#ff00ff
orchid		#da70d6
mediumvioletred		#c71585
deeppink		#ff1493
hotpink		#ff69b4
lavenderblush		#fff0f5
palevioletred		#db7093
crimson		#dc143c
pink		#ffc0cb
lightpink		#ffb6c1

Спецсимволы HTML

Имя	Вид	Описание
		Неразрывный пробел
&euro;	€	Знак евро
&sect;	§	Параграф
&deg;	°	Градус
&plusmn;	±	Плюс-минус
&frac14;	$^1/_4$	Дробь — одна четвертая
&frac12;	$^1/_2$	Дробь — одна вторая
&frac34;	$^3/_4$	Дробь — три четвертых
&times;	×	Знак умножения
&lt;	<	Знак «меньше»
&gt;	>	Знак «больше»

Свойства CSS для управления шрифтами

Свойство	Значение	Описание	Пример
font-family	имя шрифта	Задает список шрифтов	P {font-family: Arial, serif}
font-style	normal italic oblique	Нормальный шрифт Курсив Наклонный шрифт	P {font-style: italic}
font-variant	normal small-caps	Капитель (особые прописные буквы)	P {font-variant: small-caps}
font-weight	normal lighter bold bolder 100–900	Нормальная жирность Светлое начертание Полужирный Жирный 100 — светлый шрифт, 900 — самый жирный	P {font-weight: bold}
font-size	normal pt px %	Нормальный размер Пункты Пиксели Проценты	font-size: normal font-size: 12pt font-size: 12px font-size: 120%

Свойства CSS для управления видом текста

Свойство	Значение	Описание	Пример
line-height	normal множитель значение в %	Межстрочный интервал	line-height: normal line-height: 1.5 line-height: 12px line-height: 120%
text-decoration	none underline overline line-through blink	Убрать все оформление Подчеркивание Линия над текстом Перечеркивание Мигание текста	text-decoration: none
text-transform	none capitalize uppercase lowercase	Убрать все эффекты Начинать С Прописных ВСЕ ПРОПИСНЫЕ все строчные	text-transform: capitalize
text-align	left right center justify	Выравнивание текста	text-align: justify
text-indent	значение %	Отступ первой строки	text-indent: 15px; text-indent: 10%

(Название учреждения образования)

Учебный год	Имя и фамилия учащегося	Состояние учебного пособия при получении	Оценка учащемуся за пользование учебным пособием
20 /			
20 /			
20 /			
20 /			
20 /			
20 /			

Учебное издание

Котов Владимир Михайлович
Лапо Анжелика Ивановна
Быкадоров Юрий Александрович
Войтехович Елена Николаевна

ИНФОРМАТИКА

Учебное пособие для 11 класса
учреждений общего среднего образования с русским языком обучения
(с электронными приложениями)

Зав. редакцией *Г. А. Бабаева*. Редакторы *Е. И. Черникова, Д. И. Симанович*. Художественный редактор *Е. А. Проволович*. Художники *А. А. Шевченко, А. Н. Богушевич, В. Ч. Проволович*. Техническое редактирование и компьютерная верстка *И. И. Дубровской*. Корректоры *О. С. Козицкая, Е. П. Тхир, А. В. Алешко*.

Подписано в печать 09.06.2021. Формат $70 \times 90\ ^{1}/_{16}$. Бумага офсетная. Гарнитура школьная. Печать офсетная. Усл. печ. л. 8,19. Уч.-изд. л. 8,0 + 20,0 эл. прил. Тираж 118 200 экз. Заказ 947.

Издательское республиканское унитарное предприятие «Народная асвета» Министерства информации Республики Беларусь. Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий 1/2 от 08.07.2013. Пр. Победителей, 11, 220004, Минск, Республика Беларусь.

Республиканское унитарное предприятие «Издательство «Белорусский Дом печати». Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 2/102 от 01.04.2014. Пр. Независимости, 79/1, 220013, Минск, Республика Беларусь.