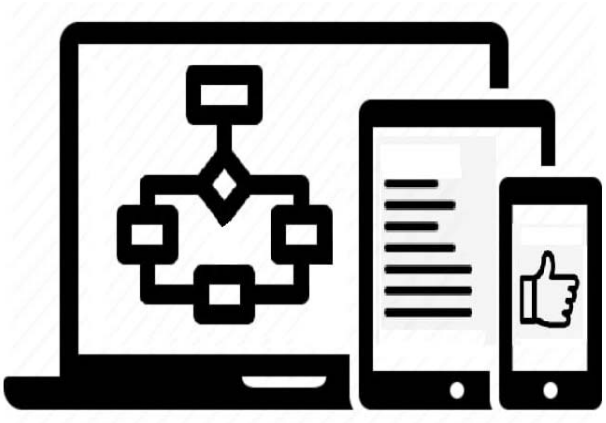
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» очной и заочной форм обучения



Могилев 2022

УДК 004.35: 004.3
ББК 32.973.202-04
К27

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «4» марта 2022 г., протокол № 9

Составители: ст. преподаватель Т. Л. Шебан;
ст. преподаватель В. М. Прудников

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации к лабораторным работам для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» очной и заочной форм обучения.

Учебно-методическое издание

КОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Ответственный за выпуск	В. В. Кутузов
Корректор	А. А. Подошевко
Компьютерная верстка	Е. В. Ковалевская

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

1 Лабораторная работа № 1. Форматирование документа с использованием стилей

Цель работы.

Научиться использовать библиотеку стилей Microsoft Word, создавать пользовательские стили, создавать оглавление, форматировать документ с использованием стилей на примере оформления пояснительной записки к курсовому проекту.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, результат выполнения работы.

Основные теоретические положения.

Стили представляют собой наборы команд форматирования. При создании стиля пользователь указывает значения отдельных параметров форматирования, которые должны быть включены в создаваемый стиль, для последующего применения всех этих параметров совместно посредством выбора имени этого стиля.

Стили определяют форматирование символов, текстовых фрагментов, абзацев, строк таблиц или уровней структуры документа. Существует два различных типа стилей:

1) стиль символа содержит параметры форматирования символов, включая шрифт, размер, начертание, положение и интервалы;

2) стиль абзаца содержит параметры форматирования абзацев, такие как межстрочные интервалы, отступы, выравнивание и позиции табуляции.

Экспресс-стили Word и инструменты для работы с ними находятся на панели «Стили» вкладки «Главная».

Для облегчения работы со стилями существует специальный механизм «инспектор стилей», который позволяет отслеживать используемые в документе стили абзаца и текста. Для вызова инспектора стилей служит кнопка с изображением буквы «А» и увеличительного стекла.

Контрольные вопросы

1 Что такое стиль?

2 Какие виды стилей существуют?

3 Как определить (переопределить) новый стиль?

2 Лабораторная работа № 2. Форматирование документа со сложной структурой

Цель работы.

Освоение основных приемов работы с графическими примитивами, надписями, формулами, расположение их в тексте документа.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, результат выполнения работы.

Основные теоретические положения.

Инструменты для работы с графикой находятся на панели «Иллюстрации» вкладки «Вставка».

Кнопка «Фигуры» служит для быстрого создания графических примитивов. Окно панели «Стили фигур» содержит расширенные параметры форматирования «Формат автофигуры». В этом окне можно произвести большинство настроек форматирования.

Если в документ вставлено несколько фигур, перекрывающих друг друга, то их относительный порядок размещения можно настроить при помощи кнопок «На передний план» и «На задний план».

Случаются такие ситуации, когда в документе размещены несколько объектов и с ними одновременно нужно произвести какие-либо действия (увеличить, уменьшить, переместить). В этом случае целесообразно произвести группировку объектов. Для группировки фигур их необходимо выделить и выполнить. Команду «Группировать» панели «Упорядочить».

Особым видом графического примитива является Надпись. Этот примитив может содержать «в себе» текст.

Такие графические элементы, содержащие текст, можно связывать между собой. Для этого их необходимо предварительно разместить в документе. Затем выделить надпись, с которой будет начинаться текст. После этого на панели «Текст» воспользоваться кнопкой «Создать связь».

Графика SmartArt позволяет быстро создавать разнообразные красочные схемы. При выборе шаблонов SmartArt необходимо учитывать их первоначальное предназначение. Для вставки объекта SmartArt служит одноименная кнопка на панели «Иллюстрации» вкладки «Вставка».

Для вставки математических формул на вкладке Вставка в группе Символы щелкните стрелку рядом с пунктом Формула, а затем выберите подходящую формулу или выберите пункт Вставить новую формулу. При вставке новой формулы появляется контекстная лента Работа с формулами и Конструктор. Конструктор содержит все необходимые инструменты для создания и редактирования формул.

Контрольные вопросы

1 Какие графические примитивы вам известны?
2 Как выполнить группировку фигур?
3 Как вставить математическую формулу?

3 Лабораторная работа № 3. Разработка таблиц в MS Access

Цель работы.
Освоение основных приемов работы с таблицами в среде MS Access.

Порядок выполнения работы.
1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить и представить отчет в виде файла в формате базы данных MS Access. Отчет должен содержать название работы, цель работы, результат выполнения работы.

Основные теоретические положения.
База данных – это реализованная с помощью компьютера модель данных, отражающая состояние объектов и их отношения. База данных (БД) предполагает наличие комплекса программных средств, обслуживающих ее и позволяющих использовать содержащуюся в ней информацию. Такие комплексы программ называют СУБД (системами управления базами данных).

В Access база данных – это файл, в котором хранятся все объекты, необходимые для обеспечения работы пользователя: таблицы, запросы, формы, отчеты, страницы доступа к данным, макросы и модули.

Все управление базой данных осуществляется в Окне базы данных, появляющемся при открытии БД. Это окно содержит основные объекты Access: таблицы, запросы, формы, отчеты, макросы и модули. В строке заголовка окна всегда отображается имя открытой или создаваемой БД.

Работать с таблицей MS Access можно в двух основных режимах: в режиме конструктора и в режиме таблицы (можно выбрать из контекстного меню).

Режим таблицы используется для просмотра, добавления, изменения, сортировки и удаления данных.

В режиме конструктора задается структура таблицы, т. е. определяются типы, свойства полей, их число и названия (заголовки столбцов). Он используется для изменения только структуры таблицы.

Между таблицами могут быть установлены связи один-к-одному, один-ко-многим. Информация о каждом типе связи имеется в справочной системе Access. Поля таблиц, используемые для установления связей должны иметь одинаковый тип данных (допускается связь данных типа счетчик с числовым типом), их значения должны отвечать требованию целостности данных в базе данных.

Во всех лабораторных работах в среде Microsoft Access используется учебная база данных Борей.mdb.

Контрольные вопросы

1 Что такое база банных?

2 Назовите способы создания таблиц.

3 Как установить связи между таблицами?

4 Лабораторная работа № 4. Создание запросов в MS Access

Цель работы.

Освоение основных приемов работы с запросами в среде MS Access.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате базы данных MS Access. Отчет должен содержать название работы, цель работы, результат выполнения работы.

Основные теоретические положения.

Запросы используются для просмотра, изменения и анализа данных различными способами. Запросы также можно использовать в качестве источников записей для форм, запросов, отчетов. В Microsoft Access есть несколько типов запросов.

Запросы на выборку возвращают данные из одной или нескольких таблиц и отображают их в виде таблицы, записи в которой можно обновлять (с некоторыми ограничениями). Запросы на выборку можно также использовать для группировки записей и вычисления сумм, средних значений, подсчета записей и нахождения других типов итоговых значений.

Запросы с параметрами – это запрос, при выполнении отображающий в собственном диалоговом окне приглашение ввести данные, например, условие для возвращения записей или значение, которое требуется вставить в поле.

Перекрестные запросы используют для расчетов и представления данных в виде таблицы по двум наборам данных, один из которых определяет заголовки столбцов, а другой заголовки строк. Перекрестный запрос создается с помощью мастера или самостоятельно в режиме конструктора запроса.

Запросы на изменение за одну операцию изменяют или перемещают несколько записей.

Запрос SQL – это запрос, создаваемый при помощи инструкций SQL. Язык SQL (Structured Query Language) используется при создании запросов, а также для обновления и управления реляционными базами данных, такими как базы данных Microsoft Access.

Контрольные вопросы

1 Что такое запрос?
2 Какие виды запросов существуют?
3 Как создать запрос на основе нескольких таблиц?

5 Лабораторная работа № 5. Разработка форм и отчетов в MS Access

Цель работы.

Освоение основных приемов работы с формами и отчетами в среде MS Access.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить и представить отчет в виде файла в формате базы данных MS Access. Отчет должен содержать название работы, цель работы, результат выполнения работы.

Основные теоретические положения.

Отчет – это гибкое и эффективное средство для организации данных при выводе на печать. С помощью отчета имеется возможность вывести необходимые сведения в том виде, в котором требуется. Больше всего сведений в отчете берется из базовой таблицы, запроса или инструкции SQL,

являющихся источниками данных для отчета. Другие сведения вводятся при разработке отчета.

Разделы отчета.

Вся информация в отчете разбивается на разделы, каждый из которых имеет специальное назначение. При печати разделы располагаются на страницах в определенном порядке.

1 Заголовок отчета.

2 Верхний колонтитул.

3 Область данных (данные из таблиц).

4 Нижний колонтитул.

5 Примечание отчета.

В режиме конструктора на экране отображается макет каждого раздела отчета в одном экземпляре. При печати некоторые разделы могут неоднократно повторяться. Элементы управления, такие как надпись или поле, находящиеся в разделе, определяют местоположение информации в отчете.

Отчет с группировкой данных позволяет вычислить итоговые значения для групп, а также представить информацию в удобном для использования виде.

Контрольные вопросы

1 Для чего используется объект базы данных «отчет»?

2 Какие разделы включает объект базы данных «форма»?

3 Какие элементы управления вам известны?

6 Лабораторная работа № 6. Базы данных в MS Excel. Фильтры

Цель работы.

Освоение основных приемов работы с базами данных в среде MS Excel.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа MS Excel. Отчет должен содержать название работы, цель работы, результат выполнения работы.

Основные теоретические положения.

Обычная двухмерная таблица Excel, созданная с соблюдением некоторых определенных правил является таблицей базы данных (списком).

Столбцы таблицы Excel – это поля базы данных, а строки – это записи базы данных.

Поле (столбец) содержит информацию об одном признаке для всех записей базы данных. Запись (строка) состоит из нескольких (по числу полей) разнообразных информационных сообщений (признаков), характеризующих один объект базы данных.

Для сложной сортировки (по нескольким столбцам) необходимо выделить ячейку внутри сортируемого списка. Затем выбрать на вкладке Данные – группа Сортировка и фильтр – Сортировка. MS Excel отобразит диалоговое окно Сортировка. Определив критерии сортировки для одного столбца, при необходимости нажать кнопку Добавить уровень, выберите имя другого столбца в поле Затем по и установите требуемые признак и порядок сортировки. Если первая строка вашего списка не содержит заголовков, выберите в поле Сортировать по имя столбца.

В Excel имеется два способа отображения записей списка, удовлетворяющих заданным условиям: автофильтр и расширенный фильтр. После выбора на вкладке Данные инструмента Фильтр в строке имен полей появляются кнопки раскрывающихся списков, содержащих команды Фильтр по цвету, Числовые фильтры, Текстовые фильтры, Фильтры по дате, а также перечень всех имеющихся значений поля.

Расширенный фильтр используется для задания сложных условий фильтрации. Чтобы его применить, нужно сначала создать диапазон критериев. Первая строка этого диапазона должна содержать имена полей, по которым будут задаваться условия, в следующих строках вводятся условия. Между критериями в одной строке идет связь по И, между строками критериев – по ИЛИ.

После создания диапазона условий указывается ячейка списка и выбирается на вкладке Данные инструмент Расширенный фильтр.

Контрольные вопросы

1 Что такое база данных Excel?
2 Какие виды сортировки базы данных существуют?
3 Как применяется расширенный фильтр?

7 Лабораторная работа № 7. Консолидация данных. Сводные таблицы и сводные диаграммы

Цель работы.
Изучение способов группировки данных и подсчета итоговых значений.

Порядок выполнения работы.
1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа MS Excel. Отчет должен содержать название работы, цель работы, результат выполнения работы.

Основные теоретические положения.

Если в разных листах книги или даже разных книгах имеются однотипные значения, которые нужно разместить в одном месте или провести с ними однотипные операции, это можно сделать при помощи функции Консолидация. При ее использовании данные из указанных источников собираются в одном месте, и любые общие значения будут объединяться, как указано. Это позволяет более наглядно анализировать итоговые данные.

Чтобы выполнить консолидацию, сначала нужно выделить первую ячейку места, в котором будут расположены консолидированные данные, затем выбрать команду Данные – Работа с данными – Консолидация.

Сводные таблицы обеспечивают очень удобный интерфейс к хранилищам данных различной сложности и разного объема. Сводная таблица – это динамическая таблица специального вида, построенная на базе одной или нескольких исходных таблиц и содержащая сводную информацию по этим таблицам. Базами данных для сводных таблиц могут быть списки, таблицы, расположенные на рабочих листах Excel, либо внешние источники данных (например, базы данных Access). При создании сводной таблицы пользователь распределяет информацию, указывая, какие элементы и в каких полях сводной таблицы будут содержаться.

Для работы в Excel со сводными таблицами существует команда Вставка – Сводная таблица. После ее активизации в появившемся окне Создание сводной таблицы нужно указать исходные данные и размещение итогов сводной таблицы.

Контрольные вопросы

1 Что такое сводная таблица?
2 Как создать и редактировать сводную таблицу?
3 Что такое консолидация данных?

8 Лабораторная работа № 8. Ввод-вывод данных. Разработка программы линейной структуры

Цель работы.

Изучение *Visual Basic for Application* на примере линейной программы, организации ввода и вывода данных с помощью диалоговых окон и с помощью ячеек листа Excel.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, схему алгоритма, код программы.

Основные теоретические положения.

Для ввода данных с листа или вывода на лист используется объект Worksheets и его методы Range или Cells.

Метод Range использует в качестве аргументов одну или две ссылки на ячейки и возвращают объект Range. Ссылки на ячейки должны быть оформлены в стиле A1 (колонка-буква-строка-число).

Например, выражение X = Worksheets(«Лист1»).Range(«B1»).Value присваивает переменной X значение ячейки «B1» листа «Лист1».

Выражение Worksheets («Лист1»). Range («B7:C9»). Value =3 выводит в диапазон ячеек «B7:C9» листа «Лист1» число 3.

Метод Cells, получая в качестве аргументов два целых числа, возвращают объект, содержащий единичную ячейку. Аргументы определяют номера строки и столбца выбранной ячейки.

Например, выражение A=Worksheets(1).Cells(1,2).Value переменной A присваивает значение из ячейки первой строки и второго столбца первого листа,

Выражением Worksheets(1).Cells(2,2).Value= X в ячейку второй строки и второго столбца заносится значение переменной X.

Для ввода данных с клавиатуры используется окно ввода InputBox, а для вывода информации на экран – окно сообщений MsgBox.

Синтаксис: InputBox(prompt[, title] [, default]),

где prompt – строковое выражение, отображаемое как сообщение в диалоговом окне;

title – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения;

default – строковое выражение, отображаемое в поле ввода как используемое по умолчанию, если пользователь не введет другую строку. Если этот аргумент опущен, поле ввода изображается пустым.

MsgBox(prompt[, buttons] [, title]),

где prompt – строковое выражение, отображаемое как сообщение в диалоговом окне;

buttons – числовое выражение, представляющее сумму значений, которые указывают число и тип отображаемых кнопок, тип используемого значка, основную кнопку и модальность окна сообщения. Значение по умолчанию этого аргумента равняется 0;

title – строковое выражение, отображаемое в строке заголовка диалогового окна. Если этот аргумент опущен, в строку заголовка помещается имя приложения.

Часто процедура MsgBox используется в «минимальном» варианте – только для вывода сообщения, с одной кнопкой – OK. В этом случае аргументы не берутся в скобки. Например, MsgBox «Значение переменной X=» & X.

Контрольные вопросы

1 Назовите способы ввода и вывода данных.
2 Назовите основные типы данных VBA.
3 Какие кнопки по умолчанию размещены на окне вывода?

9 Лабораторная работа № 9. Разработка программы разветвленной структуры

Цель работы.
Изучение оператора условия IF.

Порядок выполнения работы.
1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, схему алгоритма, код программы.

Основные теоретические положения.
Оператор ветвления позволяет выбирать и выполнять действия в зависимости от истинности некоторого условия. Имеется два варианта синтаксиса:
1) однострочная форма записи условного оператора:
IF условие Then **[**операторы 1**] [**Else операторы 2**]**
2) блочная форма записи (оператор IF расположен на нескольких строках и может проверять несколько условий):
IF условие 1 Then
[операторы 1**]**
[ElseIf условие – n Then
[операторы-n **]…**
[Else
[ИначеОператоры**]]**
End If
Здесь условие обязательно в обоих вариантах. Оно может быть числовым или строковым выражением со значениями True или False. Если условие истинно (True), выполняется последовательность «операторы 1», если ложно, то «операторы 2».

В представленном ниже примере определяется, является ли введенное число четным.

```
Public Sub chet()
Dim n As Double
n = InputBox("Введите число")
If n Mod 2 = 0 Then
  MsgBox ("Введенное число " & n & " является четным")
Else
  MsgBox ("Введенное число " & n & " не является четным")
End If
End Sub
```

Контрольные вопросы

1 Какие виды оператора IF существуют? Чем они отличаются друг от друга?
2 Сколько условий может проверять оператор IF блочной формы записи?

10 Лабораторная работа № 10. Оператор цикла с параметром For…Next. Обработка массивов

Цель работы.

Изучение оператора цикла с параметром For…Next. Освоение основных способов обработки массивов.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, схему алгоритма, код программы.

Основные теоретические положения.

Оператор цикла For позволяет повторять группу операторов заданное число раз. Синтаксис:

For *счётчик_цикла* = *начало* to *конец* [step шаг]
Тело цикла
Next [*счётчик_цикла*]

Здесь *счётчик_цикла* – это числовая переменная. В начале выполнения цикла она принимает значение, задаваемое числовым выражением *начало***.** Числовое выражение *конец* задает заключительное выражение счётчика цикла. Числовое выражение шаг не обязательно и по умолчанию=1. *Тело цикла* – это последовательность операторов, которая будет выполнена заданное число раз.

Массив – совокупность однотипных элементов данных (чисел, логических данных, символов), которой при обработке присвоено определенное имя. Массивы бывают статические и динамические.

Статическими называются массивы, количество элементов в которых заранее известно и не изменяется в ходе выполнения программы.

Динамические массивы – это массивы, в которых либо неизвестно начальное количество элементов, либо размерность массива (количество элементов) изменяется при выполнении программы.

Обращение к элементу массива осуществляется следующим образом: указывается имя массива, а затем в круглых скобках указывается номер элемента в массиве. Если массив двумерный – указывается вначале номер строки, затем через запятую номер столбца.

В представленном ниже примере объявляется двумерный динамический массив, который после ввода размерности переопределяется в соответствии с введенным количеством строк и столбцов. Далее массив заполняется случайными числами от -10 до 10 и выводится на лист Excel.

```
Public Sub massiv()
Dim A() As Double
n = InputBox("Введите количество строк массива")
m = InputBox("Введите количество столбцов массива")
ReDim A(1 To n, 1 To m)
For i = 1 To n
  For j = 1 To m
        A(i, j) = Int(Rnd() * 20 - 10)
        Cells(i, j) = A(i, j)
  Next
Next
End Sub
```

Контрольные вопросы

1 Какие виды массивов существуют? Чем они отличаются друг от друга?

2 Назовите способы заполнения массива.

11 Лабораторная работа № 11. Разработка программ циклической структуры

Цель работы.

Освоение основных приемов применения операторов цикла с предусловием и постусловием.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, схему алгоритма, код программы.

Основные теоретические положения.

Циклы данного вида используются, когда заранее неизвестно, сколько раз будет выполняться тело цикла.

Операторы циклов с предусловием: Do While…Loop, While…Wend, Do Until…Loop. Синтаксис операторов циклов с предусловием представлен в таблице 11.1.

Операторы циклов с постусловием: Do…Loop While, Do…Loop Until. Синтаксис операторов циклов с постусловием представлен в таблице 11.2.

Таблица 11.1 – Синтаксис операторов с предусловием

Синтаксис	**Do While** <*условие*> <*тело цикла*> [**Exit Do**] … **Loop**	**While** <*условие*> <*тело цикла*> **Wend**	**Do Unlil** <*условие*> <*тело цикла*> [**Exit Do**] … **Loop**
Порядок выполнения	**<***Тело цикла***>** будет выполняться в том случае, когда <*условие*> имеет значение истина (TRUE) (цикл продолжается при истинном значении <*условия***>**). Если <*условие*> ложно (FALSE), то выполняются операторы, стоящие за циклом. В первом случае есть возможность досрочного выхода из цикла (это реализовано через **Exit Do)**		**<***Тело цикла***>** выполняется до тех пор, пока <*условие*> не примет значение истина (цикл продолжается при ложном значении <*условия*>). Есть возможность досрочного выхода из цикла (это реализовано через **Exit Do**)

Отличие циклов с предусловием от циклов с постусловием заключается в том, что тело цикла первых может не выполниться ни разу, в то время как тело цикла с постусловием всегда выполнится хотя бы один раз.

Таблица 11.2 – Синтаксис операторов с постусловием

Синтаксис	**Do** <*тело цикла*> [**Exit Do**] … **Loop While** <*условие*>	**Do** <*тело цикла*> [**Exit Do**] … **Loop Until <***условие*>
Порядок выполнения	<*Тело цикла*> будет выполняться в том случае, когда <*условие*> имеет значение истина (цикл продолжается при истинном значении <*условия*>). Если <*условие*> ложно, то выполняются операторы, стоящие за циклом. Предоставлена возможность досрочного выхода из цикла (это реализовано через **Exit Do**)	**<***Тело цикла***>** выполняется до тех пор, пока <*условие*> не примет значение истина (цикл продолжается при ложном значении <*условия*>). Есть возможность досрочного выхода из цикла (это реализовано через **Exit Do**)

Представленный ниже пример демонстрирует использование цикла с предусловием. В этом примере числа вводятся, пока не встретится три подряд идущих положительных числа, после чего выводится сообщение об общем количестве введенных чисел.

```
Public Sub chet()
Dim n As Double
Dim kolp As Double
Dim kolv As Double
Do While kolp < 3
n = InputBox("Введите число")
If n > 0 Then
kolp = kolp + 1
Else
kolp = 0
End If
kolv = kolv + 1
Loop
MsgBox ("Всего введено " & kolv & " чисел")
End Sub
```

Контрольные вопросы

1 В каких случаях используются циклы DO?
2 Назовите виды циклов DO. Чем они отличаются?

12 Лабораторная работа № 12. Обработка строковых данных

Цель работы.
Изучение функций обработки строк.

Порядок выполнения работы.
1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, схему алгоритма, код программы.

Основные теоретические положения.
Строка – упорядоченная последовательность символов. Каждый символ строковой величины занимает 1 байт памяти (код ASCII). Количество символов в строке называется ее длиной.

Строковая переменная описывается в разделе описания переменных:

Dim <идентификатор> As String, например:

Dim Name As String

Операции над строками.

Операция сцепления (конкатенации) & применяется для соединения нескольких строк в одну результирующую строку. Сцеплять можно как строковые константы, так и переменные.

В VBA имеются следующие функции обработки строковых выражений, представленные в таблице 12.1.

Таблица 12.1 – Функции обработки строковых выражений

Функция	Возвращаемое выражение
Asc	Возвращает ASCII-код начальной буквы строки. Синтаксис: Asc(Строка)
Chr	Преобразует ASCII-код в строку. Синтаксис: Chr(Код) Например, Chr (13) – переход на новую строку
Lcase	Преобразует строку к нижнему регистру. Синтаксис: Lcase(Строка)
Ucase	Преобразует строку к верхнему регистру. Синтаксис: Ucase(Строка)
Left	Возвращает подстроку, состоящую из заданного числа первых символов исходной строки. Синтаксис: Left(string, length) Аргументы: string – исходная строка, length – число символов
Right	Возвращает строку, состоящую из заданного числа последних символов исходной строки. Синтаксис: Right(string, length) Аргументы: string – исходная строка, length – число символов
Mid	Возвращает подстроку строки, содержащую указанное число символов. Синтаксис: Mid(string, start [, length]) Аргументы: string – строковое выражение, из которого извлекается подстрока, start – позиция символа в строке string, с которого начинается нужная подстрока, length – число возвращаемых символов подстроки
Len	Возвращает число символов строки. Синтаксис: Len(Строка)
LTrim	Возвращает копию строки без пробелов в начале. Синтаксис: LTrim(Строка)
Rtrim	Возвращает копию строки без пробелов в конце. Синтаксис: RTrimfСтрока)
Trim	Возвращает копию строки без пробелов в начале и в конце. Синтаксис: Trim(Строка)
Space	Возвращает строку, состоящую из указанного числа пробелов. Синтаксис: Space(Число)

Окончание таблицы 12.1

Функция	Возвращаемое выражение
String	Возвращает строку, состоящую из указанного числа повторений одного и того же символа. Синтаксис: String(number,character) Аргументы: number – число повторений символа, character – повторяемый символ
InStr	Возвращает позицию первого вхождения одной строки внутри другой строки. Синтаксис: InStr([start,]stringl, string2[, compare]) Аргументы: start – числовое выражение, задающее позицию, с которой начинается каждый поиск. Если этот аргумент опущен, поиск начинается с первого символа строки, stringl – строковое выражение, в котором выполняется поиск, string2 – искомое строковое выражение, compare – указывает способ сравнения строк. Допустимые значения: 0 (для двоичного сравнения), 1 (посимвольное сравнение без учета регистра)

13 Лабораторная работа № 13. Разработка программ с использованием подпрограмм и функций

Цель работы.

Разработка программ с использованием подпрограмм и функций.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, схему алгоритма, код программы.

Основные теоретические положения.

Подпрограмма – программа, реализующая вспомогательный алгоритм. Основная программа – программа, реализующая основной алгоритм решения задачи и содержащая в себе обращения к подпрограммам. В VBA существуют два типа подпрограмм: подпрограммы-функции и подпрограммы-процедуры.

Отличие функции от процедуры заключается в том, что результатом исполнения операторов, образующих тело функции, всегда является некоторое единственное значение, поэтому обращение к функции можно использовать в соответствующих выражениях наряду с переменными и константами.

Подпрограмма-процедура имеет следующий синтаксис:

[Private | Public] Sub <имя процедуры> ([<список аргументов>])

[<Инструкции>]
[Exit Sub]
[<Инструкции>]
End Sub,

где Public указывает, что процедура Sub доступна во всех других процедурах во всех модулях;

Private указывает, что процедура Sub доступна для других процедур только для того модуля, в котором она описана;

Sub, End Sub – служебные слова VBA;

<имя процедуры> – имя процедуры, удовлетворяющее стандартным правилам именования;

<список аргументов> – список переменных, представляющих аргументы, которые передаются в процедуру Sub при её вызове. Имена переменных разделяются запятой;

<Инструкции> – любой набор команд VBA;

Exit Sub – инструкция, выполнение которой приводит к выходу из процедуры.

Синтаксис элемента <список аргументов>:

[ByVal | ByRef] <Имя переменной> [As <Тип>],

где ByVal – ключевое слово, указывающее, что аргумент передается по значению;

ByRef – ключевое слово, указывающее, что аргумент передается по ссылке. Описание ByRef используется в VBA по умолчанию;

<Имя переменной> – имя переменной, удовлетворяющее стандартным правилам именования переменных;

<Тип> – тип данных аргумента, переданного в процедуру.

В качестве результата процедура может возвращать в вызывающую программу множество простых или структурированных величин или не возвращать никаких значений. Среди параметров процедуры указываются как аргументы, так и результаты.

Обращение к процедуре – отдельный оператор.

Вызов процедуры из другой процедуры можно произвести несколькими способами.

Первый способ.

<Имя процедуры> <Список фактических параметров>

<Имя процедуры> – имя вызываемой процедуры;

<Список фактических параметров> – список аргументов, передаваемых процедуре; он должен соответствовать списку, заданному в процедуре, по количеству и типу.

Второй способ.

Call <Имя процедуры> (<Список фактических параметров>)

Call – служебное слово VBA;

<Имя процедуры> – имя вызываемой процедуры;

<Список фактических параметров> – список аргументов, передаваемых процедуре; он должен соответствовать списку, заданному в процедуре, по количеству и типу.

Заметим, что при втором способе вызова процедуры в отличие от первого список фактических параметров должен быть заключен в круглые скобки; в качестве разделителя в списке используется запятая.

Подпрограмма-функция имеет следующий синтаксис.

[Private | Public] Function <имя функции> [(<список аргументов>)]
[<Инструкции>]
[Exit Function]
[<Инструкции>]
End Sub

Синтаксис инструкции Function содержит те же элементы, что и Sub.

Тип функции может быть только простым типом. Для возврата значения из функции следует присвоить значение имени функции.

Обращение к функции является операндом в выражении. Подпрограмма-функция вызывается в выражении по своему имени, за которым следует вписок аргументов в скобках.

При обращении к подпрограмме происходит передача ей аргументов по ссылке (если формальный параметр является параметром-переменной, описан как ByRef) или по значению (является параметром-значением, описан как ByVal).

Если параметр определен как параметр-значение (с помощью ключевого слова ByVal), то перед вызовом подпрограммы это значение вычисляется, полученный результат копируется во временную память и передается подпрограмме. Любые возможные изменения в подпрограмме параметра-значения никак не воспринимаются вызывающей подпрограммой, т. к. в этом случае изменяется копия фактического параметра.

Если параметр определен как параметр-переменная (по умолчанию или с помощью ключевого слова ByRef), то при вызове подпрограммы передается сама переменная, а не ее копия. Изменение параметра-переменной приводит к изменению самого фактического параметра в вызывающей подпрограмме.

Если в качестве фактического параметра используется константа, транслятор блокирует любые присваивания константе нового значения в теле подпрограммы.

Контрольные вопросы

1 Что такое подпрограмма?
2 Чем отличаются подпрограмма-функция и подпрограмма-процедура?
3 Как происходит передача параметров в подпрограмму?

14 Лабораторная работа № 14. Разработка пользовательских форм

Цель работы.

Научиться создавать пользовательские формы; создавать и использовать элементы управления; ознакомиться с их свойствами и методами.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание лабораторных работ № 8 и 9 с использованием ввода и вывода на пользовательскую форму.

Форма должна содержать элементы управления TextBox – текстовые поля для ввода и вывода данных, Label – надписи, CommandButton – кнопки для выполнения вычислений и очистки текстовых полей, Image – рисунок с условием задачи. Выполнить проверку на ввод нечисловых значений в текстовые поля с использованием функции IsNumeric.

3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, код программы.

Основные теоретические положения.

Окно проекта в редакторе VBA активизируется выбором команды View – Project Explorer.

В окне проекта представлена иерархическая структура файлов, форм и модулей текущего проекта. В проекте автоматически создается модуль для каждого рабочего листа и для всей книги. Кроме того, модули создаются для каждой пользовательской формы макросов и классов.

Для создания диалоговых окон, разрабатываемых приложений в VBA, используются формы. Редактор форм является одним из основных инструментов визуального программирования. Форма в проект добавляется с помощью команды Insert – UserForm. В результате на экран выводится незаполненная форма с панелью инструментов Toolbox.

Используя панель инструментов из незаполненной формы, можно сконструировать любое требуемое для приложения диалоговое окно, перетащив на форму нужные элементы управления.

В окне свойств перечисляются основные установки свойств выбранной формы или элемента управления. Используя это окно, можно просматривать свойства и изменять их установки. Окно свойств активируется командой View – Property Window.

Контрольные вопросы

1 Назовите основные элементы управления формы.

2 Назовите основные свойства элементов управления формы.

15 Лабораторная работа № 15. Разработка программ для работы с текстовыми данными

Цель работы.

Освоение основных приемов работы с текстовыми данными и файлами.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Выполнить задание, полученное у преподавателя.

3 Оформить и представить отчет в виде файла в формате документа Word. Отчет должен содержать название работы, цель работы, задание, схему алгоритма, код программы.

Основные теоретические положения.

VBA позволяет обрабатывать файлы, работать с дисками и каталогами посредством объектной файловой системы (FSO – File System Object Model);

FSO предоставляет разработчику возможность работы с файлами в ясной объектно-ориентированной манере. FSO опирается на библиотеку Microsoft Scripting Runtime Library (файл Scrrun.dll), которая совместно используется несколькими средами разработки и позволяет, в частности, расширить функции написания сценариев.

Для использования этой библиотеки надо установить ссылку на Microsoft Scripting Runtime в диалоговом окне References – VBAProject, которое отображается на экране выбором команды Tools – References. После этого можно приступить к программированию объектов, функционирование которых обеспечивается данной библиотекой.

Основными объектами FSO для работы с файлами являются File, который является элементом семейства Files, и имеет два свойства Count и Item, и TextStream, который позволяет работать с файлами.

Для получения доступа к существующему файлу (объекту File) надо воспользоваться методом GetFile объекта FileSystemObject.

Синтаксис: GetFile (filespec)

Прежде чем получать доступ к каталогу, желательно убедиться, что он существует. Это можно сделать при помощи метода FileExists объекта FileSystemObject.

Синтаксис: FileExists(folderspec)

Метод FileExists возвращает True, если диск существует, и False – в противном случае.

Если текстовый файл не существует, то его можно создать при помощи метода CreateTextFile объекта FileSystemObject. Этот метод создает объект TextStream и специфицирует имя создаваемого текстового файла. Объект Textstream используется для чтения и записи данных в файл.

Синтаксис: CreateTextFile(filename [, overwrite [, unicode]]),
где filename – строковое выражение, идентифицирующее создаваемый текстовый файл;

overwrite – параметр, принимающий значение равное True, если указанный файл уже существует и его следует перезаписать, и False, если существующий файл не перезаписывается;

unicode – определяет код создаваемого текстового файла, если значение параметра равно True, то файл создается как Unicode-файл, если значение параметра равно False, то файл создается как ASCII-файл.

После получения доступа к файлу возможен сбор информации о файле, работа с ним (копирование, удаление, перемещение), а также запись и считывание данных из файла при помощи следующих свойств и методов объекта File.

Delete – удаляет файл, Copy – копирует файл, Move – перемещает файл, OpenAsTextStream – создает объект Textstream и специфицирует имя открываемого текстового файла Этот объект используется для чтения, записи и добавления данных в файл:

OpenAsTextStream ([iomode, [format]]),
где iomode – задает тип операции, производимой над файлом. Допустимые значения:

ForReadmg или 1 – для чтения;

ForWntmg или 2 – для записи;

ForAppending или 8 – для добавления данных.

Метод OpenTextFile создает объект Textstream и специфицирует имя открываемого текстового файла. Этот объект используется для чтения и добавления данных в файл.

Синтаксис OpenTextFile(filename[, iomode[, create[, format]]])),
где filename – строковое выражение, идентифицирующее открываемый текстовый файл;

iomode – задает тип операции, производимой над файлом. Допустимые значения:

ForReadmg или 1 – для чтения;

ForAppendmg или 8 – для добавления данных;

create – логический параметр, который указывает, надо ли создавать новый файл, если файл с данным именем не существует;

format – задает тип файла.

Для чтения, записи и добавления данных в текстовый файл необходимо создать объект Textstream, ассоциированный с этим файлом, указать совершаемую над ним операцию (запись, добавление данных, чтение), а потом при помощи методов и свойств этого объекта реализовать требуемые операции.

Метод Read(characters) считывает указанное число символов из файла и возвращает результирующую строку.

Метод ReadLine считывает целую строку из файла и возвращает результирующую строку.

Метод ReadAll считывает все содержимое файла и возвращает результирующую строку. Используется для небольших файлов.

Метод Skip (characters) пропускает указанное число символов при считывании данных из файла.

Метод SkipLme пропускает одну строку при считывании данных из файла.

Метод Write (string) записывает в файл указанную строку, причем в конце строки размещается символ перехода на новую строку.

Метод Close закрывает открытый Textstream. По завершении работы с файлом никогда не забывайте его закрывать.

Свойство Line, Column возвращают номер строки и столбца, в котором находится текущий символ.

Свойство AtEndOfStream возвращает True, если указатель достиг конца файла.

Свойство AtEndOfLme возвращает True, если указатель достиг конца строки.

Пример записи в текстовый файл Table.txt, создаваемой в программе таблицы умножения, представлен ниже.

```
Sub MultTable()
   Dim fso As FileSystemObject
   Dim ts As TextStream
  Dim i as Integer, j as Integer
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set ts = fso.OpenTextFile("C:\Table.txt", ForWriting, True)
  For i = 1 To 9
     For j = 1 To 9
        ts.Write (Format(i * j, "00"))
        ts.Write (Space(1))
     Next j
     ts.WriteLine ' Переход на новую строку
  Next i
  ts.Close
End Sub
```

После того как текстовый файл Table.txt с таблицей умножения создан, построчно вывести его содержимое в окно Immediate можно, например, следующим кодом:

```
Sub ReadIt()
   Dim fso As FileSystemObject
   Dim ts As TextStream
   Set fso = CreateObject("Scripting.FileSystemObject")
   Set ts = fso.OpenTextFile("C:\Table.txt", ForReading)
   Do While Not ts.AtEndOfStream
      Debug.Print ts.ReadLine
   Loop
   ts.Close
End Sub
```

16 Лабораторная работа № 16. Работа в Internet

Цель работы.

Изучение и повторение терминов и понятий, изучение интерфейса и настроек браузеров, а также возможностей глобальной сети Internet.

Порядок выполнения работы.

1 Ознакомиться со списком литературы по дисциплине.

2 Изучить и выписать в конспект состав меню браузеров MS Internet Explorer и Mozilla Firefox (либо другого).

3 Найти в Internet или в литературе различные варианты определений (минимум из двух источников) следующих терминов и понятий:

– Internet;
– адрес IP v.4;
– маска IP-адреса;
– адрес IP v.6;
– DNS (*Domain Name System*);
– DNS (*Domain Name Service*);
– WWW (*World Wide Web*);
– сайт;
– HTML (*Hypertext Markup Language*);
– CSS (*Cascading Style Sheets*).

4 Оформить отчет с найденными определениями (со ссылками на источники) в виде документа MS Word с гиперссылками на них по документу. При оформлении отчета использовать возможности MS Word по форматированию текстового документа.

Контрольные вопросы

1 Структура Internet.
2 Состав служб (сервисов) Internet.
3 Состав FQDN (Fully Qualified Domain Name).
4 Структура DNS (Domain Name System).

5 Назначение HTML.

6 Как в браузере:

– настроить параметры соединения с Internet;

– просмотреть HTML-код страницы;

– изменить кодировку для интерпретации страницы;

– установить домашнюю страницу;

– просмотреть список загрузок;

– задать стили пользователя (форматирование пользователя)?

17 Лабораторная работа № 17. HTML. Оформление web-страницы средствами HTML

Цель работы.

Освоение синтаксиса HTML. Изучение текстового оформления web-страниц, порядка создания и использования списков и таблиц. Научиться вставлять изображения в web-страницы.

Порядок выполнения работы.

1 Ознакомиться с основами HTML.

2 Выполнить задание, полученное у преподавателя.

3 Оформить отчет в виде html-файла.

Основные теоретические положения.

Web-страницы (документы) используют язык HTML (Hypertext Markup Language – язык разметки гипертекста).

HTML-документ – это файл, содержащий обыкновенный текст, структурно размеченный (разметкой). Такой файл может быть подготовлен в произвольном текстовом редакторе (существуют специальные программы-конверторы и HTML-редакторы).

HTML-документ состоит из содержимого и команд, задающих структуру (разметка).

Каждая управляющая конструкция HTML-документа (тег) должна заключаться в угловые скобки: <тег>. Чаще всего в документе встречаются парные теги (открывающий и соответствующий ему закрывающий), т. к. браузеру необходимо знать область действия тега.

Открывающий и закрывающий теги называются одинаково и отличаются друг от друга только символом «наклонная» черта» или «слэш» – «/», который ставится сразу после открывающей угловой скобки закрывающего тега. Закрытие парных тегов выполняется так, чтобы соблюдались правила вложения.

```
<b><i>На этот текст воздействуют два тега</i></b>
```

Кроме того, тег может включать атрибут (атрибуты), дающий дополнительную информацию браузеру, т.е. уточняющий действие тега.

Атрибуты представляют собой дополнительные ключевые слова, отделяемые от ключевого слова, определяющего тег, и от других атрибутов пробелами и размещаемые до завершающего тег символа ">". Значение атрибута отделяется от ключевого слова атрибута символом "=" (знак равенства) и заключается в кавычки.

<h1 align="left">

17.1 Списки

В HTML используются списки следующих видов:
– *упорядоченные* (нумерованные);
– *неупорядоченные* (ненумерованные);
– *определений.*

Перед пунктами неупорядоченных списков (таблица 17.1) обычно ставятся символы-маркеры (bullets), например, точки, ромбики и т. п., в то время как пунктам упорядоченных списков предшествуют их номера.

Таблица 17.1 – Теги списков

Тег	Назначение
** – **	Создает неупорядоченный список
** – **	Создает упорядоченный список
** – **	Создает пункт списка внутри тегов **ol** или **ul**
<dl> – </dl>	Открывает и закрывает список определений
<dt> – </dt>	Создает термин в списке определений внутри элемента **dl**
<dd> – </dd>	Создает определение термина внутри элемента **dl**

17.2 Таблицы

Таблица состоит из строк и столбцов ячеек, которые могут содержать текст и рисунки. С помощью таблиц удобно верстать макеты страниц, расположив нужным образом фрагменты текста и изображений.

Для добавления таблицы на web-страницу используется тег-контейнер **<table>.**

<table> (от англ. *table* – таблица) служит контейнером для элементов, определяющих содержимое таблицы. Любая таблица состоит из строк и ячеек, которые задаются с помощью элементов **<tr>** и **<td>**. Внутри **<table>** допустимо использовать следующие элементы: **<caption>**, **<col>**, **<colgroup>**, **<tbody>**, **<td>**, **<tfoot>**, **<th>**, **<thead>** и **<tr>**.

Для добавления строк используются теги **<tr>** и **</tr>**. Чтобы разделить строки на колонки применяются теги **<td>** и **</td>.**

Для изменения вида и свойств таблицы использовались атрибуты (таблица 17.2), которые добавлялись в тег **<table>.**

<table атрибут1=... атрибут 2=...>

Таблица 17.2 – Атрибуты таблицы и их значения

Атрибут	Значение	Описание	Пример
align=	**left right center**	Выравнивание таблицы	`align="center"`
background=	**URL**	Фоновый рисунок	`background="pic.gif"`
bgcolor=	**#rrggbb**	Цвет фона таблицы	`bgcolor="#FF9900"`
border=	**n**	Толщина рамки в пикселах	`border="2"`
cellpadding=	**n**	Расстояние между ячейкой и ее содержимым	`cellpadding="7"`
cellspacing=	**n**	Дистанция между ячейками	`cellspacing="3"`
colspan=	**n**	Число ячеек, объединяемых по горизонтали	`colspan="2"`
rowspan=	**n**	Число ячеек, объединяемых по вертикали	`rowspan="3"`
nowrap		Запрещает переносы строк в тексте	`<table nowrap>`
valign=	**top bottom**	Выравнивание по высоте	`valign="top"`
width=	**n, n%**	Минимальная ширина таблицы, в пикселах или процентах	`width="90%"`
height=	**n, n%**	Минимальная высота таблицы, в пикселах или процентах	`height="18"`

Контрольные вопросы

1 Что называют ***разметкой***?
2 Приведите примеры ***парных*** и ***непарных*** тегов.
3 В каких единицах измерения задается значение атрибута ***size***?
4 При использовании какого тега появляются ***вертикальные отступы***?
5 Чем отличаются ***абсолютные*** и ***относительные*** ссылки?
6 Какие существуют ***виды*** списков?
7 С помощью каких атрибутов можно изменить ***порядок*** нумерации списка?

18 Лабораторная работа № 18. HTML. Создание и применение форм

Цель работы.
Научиться использовать формы при создании web-страниц.

Порядок выполнения работы.
1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить отчет в виде html-файла.

Основные теоретические положения.

HTML-формы предназначены для пересылки данных от удаленного пользователя к web-серверу. С их помощью можно организовать простейший диалог между пользователем и сервером, например, регистрацию пользователя на сервере или выбор нужного документа из представленного списка. Формы поддерживаются всеми популярными браузерами.

Итак, пользователь открыл web-страницу, на которой располагаются элементы управления для ввода данных. Все они объединены в общую совокупность, называемую формой. Каждая форма обладает кнопкой, щелчок по которой приводит к передаче введенных пользователем данных обрабатывающей программе. Эта программа размещается на web-сервере, который обслуживает данную страницу.

Теги, используемые при создании форм.

<button> – создаёт на веб-странице кнопку.

<fieldset> – предназначен для группирования элементов формы.

<form> – устанавливает форму на веб-странице.

<input> – позволяет создавать разные элементы интерфейса.

<keygen> – используется для генерации пары ключей – закрытого и открытого.

<label> – устанавливает связь между определённой меткой и элементом формы.

<legend> – создаёт заголовок группы элементов формы.

<optgroup> – представляет собой контейнер, внутри которого располагаются элементы <option>, объединённые в одну группу.

<option> – определяет отдельные пункты списка, создаваемого с помощью контейнера <select>.

<select> – создаёт раскрывающийся список.

<textarea> – создаёт поле для многострочного текста.

18.1 Атрибуты тега **<form>**

Форма в HTML-коде ограничивается тегами **<form>** и **</form>**, которые создают контейнер для размещения элементов управления. Элементы управления могут размещаться в таблице, которая, в свою очередь, полностью расположена в форме. Тег **<form>** не создает какой-либо отображаемой структуры. Он предназначен для внутренней группировки объектов.

Тег **<form>** обладает целым рядом параметров (атрибутов), которые задают свойства создаваемой формы.

Атрибут **action** является обязательным. Его значение – URL, указывающий на расположение того CGI-приложения, которое будет обрабатывать данные, введенные пользователем с помощью элементов управления искомой формы.

Атрибут **method** предназначен для указания способа, которым данные передаются обрабатывающему приложению. В качестве значения атрибута используется одно из двух предустановленных ключевых слов: **get** или **post**. По умолчанию используется значение **get**.

Атрибут **enctype** указывает тип данных, передаваемых из формы.

Атрибут **accept** описывает типы передаваемых файлов. Обычно не используется, т. к. сервер вполне в состоянии сам корректно распознать тип принимаемого файла.

Атрибут **name** определяет уникальное имя формы. Естественно, на одной web-странице может находиться несколько форм. В этом случае значения атрибутов **name** у них не должны совпадать.

18.2 Атрибуты тега **<input>**

Атрибут **type** является ключевым атрибутом. Его значение определяет тип создаваемого элемента управления. В качестве значения используется одно из предустановленных ключевых слов: ***text***, ***password***, ***checkbox***, ***radio***, ***submit***, ***reset***, ***file***, ***hidden***, ***image***, ***button***. По умолчанию используется значение ***text***.

```
<input type="password">
```

Атрибут **name** применяется для установки уникальных имен элементов управления. Несмотря на то, что этот атрибут не является обязательным, настоятельно рекомендуется использовать его. В сопроводительной документации CGI-приложений обязательно указывается, какие имена должны быть у соответствующих элементов управления.

Атрибут **value** используется для указания значения, отображаемого по умолчанию для кнопок и полей ввода текста. Если используются радиокнопки и независимые переключатели, то значение атрибута `value` не будет видно пользователю, но именно это значение получит обрабатывающее CGI-приложение, если пользователь выберет соответствующую радиокнопку или независимый переключатель.

Атрибут **cheked** применяется только для независимых переключателей и радиокнопок. Он устанавливает их начальное состояние. Если этот атрибут будет введен в тег **<input>**, то радиокнопка или независимый переключатель будут переведены во включенное состояние. Атрибут не имеет значений.

Атрибут **disabled** делает элемент управления недоступным для пользователя. У атрибута нет значений.

Атрибут **readonly** применяется только для полей ввода ***text*** и ***password***. Использование этого параметра означает, что данные, отображаемые в этих полях, нельзя будет изменять.

Атрибут **size** обычно задает размеры элемента управления. Но для каждого отдельного типа элементов управления его действие специфично.

Атрибут **maxlength** позволяет устанавливать максимально возможное число символов, которые пользователь может ввести в поля текстового ввода. Значение атрибута – целое положительное число.

Атрибут **src** используется в тех случаях, когда создаются элементы управления, связанные с графикой. Значением данного атрибута является URL графического файла, который содержит отображаемый рисунок.

Атрибут **alt** позволяет вставлять описание создаваемого элемента управления. Это описание может отображаться в виде маленькой подсказки, когда пользователь наводит курсор мыши на этот управляющий элемент.

Атрибут **accesskey** позволяет указать «горячую клавишу», при нажатии на которую фокус ввода переходит к данному элементу управления.

Контрольные вопросы

1 Каким тегом создаётся форма?
2 Перечислите атрибуты тега **<form>.**
3 Перечислите атрибуты тега **<input>**.
4 Как создать поле для многострочного текста в форме?
5 Как создать заголовок группы элементов формы?

19 Лабораторная работа № 19. CSS (Cascading Style Sheets)

Цель работы.

Изучение порядка создания и использования CSS при разработке web-страниц.

Порядок выполнения работы.

1 Ознакомиться с основами применения CSS.
2 Выполнить задание из лабораторной работы № 17 с использованием CSS.
3 Использовать ***все способы*** подключения CSS и ***все типы*** селекторов.
4 Сделать вывод о размере кода HTML с применением CSS, а также о целесообразности применения CSS в конкретном случае.
5 Оформить отчет в виде html-файла и двух файлов CSS.

Основные теоретические положения.

CSS (Cascading Style Sheets, Каскадные Таблицы Стилей) – это набор правил форматирования web-страницы, который может быть применен к различным элементам страницы.

В HTML для присвоения какому-либо элементу определенных свойств (таких как цвет, размер, положение на странице и т. п.) приходится каждый раз описывать эти свойства.

Применяя CSS, можно один раз описать свойства и их значения, и определить это описание как *стиль*, а в дальнейшем просто указывать, что элемент, который надо оформить соответствующим образом, должен принять эти свойства стиля (таблица 19.1).

Описание стиля можно сохранить не в тексте страницы, а в отдельном файле – это позволит использовать описание стиля на любом количестве web-страниц.

Описания стилей в web-странице должны находиться в тегах **<style></style>**, которые размещаются между тегами **<head></head>**.

`Example.css` – это CSS-файл, содержащий описание применяемых стилей. Создается CSS-файл в любом текстовом редакторе, например, в Блокноте, нужно только изменить расширение текстового файла на CSS. В CSS-файле не должны указываться теги **<style></style>**.

Можно определить стиль для любого тега отдельно. Для этого нужно в тег добавить атрибут **style=""** и описать его стиль в кавычках.

Следующий пример отображает слово «Пример» шрифтом Verdana, размером 150 % и красным цветом.

```
<h3 style=" font-size:150%; color:red">Пример</h3>
```

Таблица 19.1 – Свойства CSS

Свойство	Назначение
`font-family`	Используется для указания шрифта или шрифтового семейства, которым будет отображаться элемент. Пример: `p {font-family: Verdana, sans-serif;}`
`font-weight`	Определяет степень насыщенности шрифта: bold|bolder|lighter|normal|100|200|300|400|500|600|700|800|900 Пример: `b {font-weight: bolder;}`
`font-size`	Устанавливает размер шрифта. Параметр может указываться в процентах, пикселях или сантиметрах. Пример использования: `h1 {font-size: 200%;}`
`text-decoration`	Устанавливает эффекты оформления шрифта, такие как подчеркивание или зачеркивание текста. Пример использования: `h4 {text-decoration: underline;} (подчеркивание)`
`text-align`	Определяет выравнивание элемента. Пример: `p {text-align: left} (выравнивание по левому краю)` `p {text-align: right} (выравнивание по правому краю)`
`text-indent`	Устанавливает отступ первой строки текста. Чаще всего используется для создания параграфов с табулированной первой строкой. Пример использования: `h1 {text-indent: 60pt;}`

Окончание таблицы 19.1

Свойство	Назначение
`line-height`	Управляет интервалами между строками текста. Пример: `p {line-height: 50 %}`
`color`	Определяет цвет элемента. Пример использования для тега `h3`: `h3 {color: #0000FF;}`

`background-color`	Устанавливает цвет фона для элемента. Пример использования:
	`<h3 style="background-color:gold; color:brown;">`
	`Пример </h3>`

`margin-left`
`margin-right`
`margin-top`
`margin-bottom`
	Устанавливают значения отступов вокруг элемента.
	Пример использования для рисунка:
	`img { margin-left: 20pt}`
	`img { margin-right: 20pt}`

Контрольные вопросы

1 Какова структура правила CSS?
2 Что такое стиль и таблица стилей?
3 Способы подключения таблиц стилей CSS.
4 Какие существуют типы селекторов?
5 Для чего используется группирование в CSS?
6 У какого селектора комбинатор #?

20 Лабораторная работа № 20. CSS. Каскадирование

Цель работы.
Изучение механизма каскадирования в CSS.

Порядок выполнения работы.
1 Подготовить web-документ (за основу можно взять предыдущие работы), при форматировании которого применяется механизм каскадирования.
2 В применяемых CSS механизм каскадирования пояснить комментариями.
3 Оформить отчет в виде файла *.html и двух файлов *.css.

Основные теоретические положения.
В соответствии со спецификацией CSS имеется несколько способов подключения таблиц стилей к документу. И может получиться, что возникнет конфликт – на одно свойство конкретного элемента претендуют несколько разных значений, пришедших из разных правил (таблиц стилей).

Для разрешения конфликтов значений свойств в CSS определен механизм (порядок, алгоритм) каскадирования:

– по важности (явной приоритетности);
– по источнику;
– по специфичности;
– по расположению.

20.1 Важность (явная приоритетность)

Важность объявления настолько велика, что перевешивает все остальные факторы. CSS называет их (по вполне понятным причинам) *важными объявлениями (important declarations)* и предоставляет возможность отмечать их путем введения в объявление ключевого слова **!important** прямо перед завершающей точкой с запятой:

```
p.dark {color: #333 !important;
        background: white;}
```

Необходимо правильно размещать **!important**, иначе объявление будет признано недействительным. Ключевое слово **!important** всегда располагается *в конце объявления*, прямо перед точкой с запятой. Это особенно важно, когда дело доходит до свойств (например, **font**), значения которых могут состоять из нескольких ключевых слов:

```
p.light (color: yellow;
        font: smaller Times, serif !important;}
```

Если бы **!important** было расположено где-либо в другом месте объявления **font**, то все объявление было бы признано недействительным и ни один из его стилей не был бы применен.

Объявления, отмеченные как **!important**, не имеют особого значения специфичности, они рассматриваются отдельно от остальных. Фактически все объявления **!important** группируются вместе и тогда уже их конфликты специфичностей разрешаются относительно друг друга. Аналогично группируются все остальные объявления и конфликты свойств разрешаются с помощью специфичностей.

20.2 Источник правила

Существует *три* возможных источника правил: ***автор***, ***читатель*** и ***браузер*** пользователя. Причем на их приоритетность влияет **инструкция !important**. Таким образом, с точки зрения *приоритетности* объявлений выделяют *пять* уровней. В порядке ***уменьшения приоритетности*** это:

1) важные объявления пользователя;
2) важные объявления автора;
3) обычные объявления автора;
4) обычные объявления пользователя;
5) объявления браузера пользователя.

20.3 Специфичность

Для каждого правила браузер пользователя вычисляет специфичность (*specificity*) *селектора* и прикрепляет ее к каждому *объявлению* правила.

Специфичность селектора определяется компонентами селектора.

Значение специфичности состоит из четырех частей: **0, 0, 0, 0** и разбивается на четыре уровня: ***a***, ***b***, ***c*** и ***d***:

– если стиль встроенный (*inline*), ***a*** = 1;

– значение ***b*** равно общему количеству селекторов идентификаторов;

– значение ***c*** равно количеству классов, псевдоклассов и селекторов атрибутов;

– значение ***d*** равно количеству селекторов типов и псевдоэлементов.

20.4 Порядок расположения

Если два правила имеют совершенно одинаковые *приоритетность* и *специфичность*, тогда побеждает то из них, которое расположено в таблице стилей *позже*.

С этих позиций принимается, что стили, определенные в атрибуте **`style`** элемента, находятся в самом конце таблицы стилей документа, т. е. размещаются после всех остальных правил и поэтому имеют более высокую *специфичность*, чем любой селектор таблицы стилей.

Контрольные вопросы

1 Когда применяется механизм каскадирования?
2 Как задать явную приоритетность?
3 По какому компоненту правила вычисляется специфичность?
4 Какие существуют источники правил?
5 Как подключить пользовательский стиль в браузере?

21 Лабораторная работа № 21. CSS. Модель визуального форматирования

Цель работы.

Изучение возможностей модели визуального форматирования в CSS и основных типов позиционирования в CSS.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить отчет в виде html-файла.

21.1 Типы отображения элементов

CSS-свойство **display** задаёт тип отображения элемента. На русский язык его можно перевести фразой: «Веди себя как …». То есть, берём любой элемент **html** и говорим ему: веди себя как… блок, строка, таблица или вообще как будто тебя нет.

Наиболее часто используемые значения свойства **display**:

– none;
– block;
– inline;
– inline-block;
– list-item.

Свойство **display** изменяет поведение элемента, но не классовую принадлежность.

21.2 Замещаемые и незамещаемые элементы

CSS определяется элементами, но не все элементы создаются одинаково. Например, изображения и абзацы – это элементы разных типов, так же как **** и ** **. В CSS элементы разделяются на *замещаемые* и *незамещаемые*.

Замещаемыми (replaced) называются те элементы, содержимое которых замещается чем-то, что не содержится непосредственно в документе. Наиболее очевидный пример – элемент **img**, замещаемый файлом изображения, который является внешним по отношению к документу. Кстати, **img** фактически не имеет содержимого, как видно из простого примера:

```
<img src="howdy.gif" />
```

Основная масса элементов HTML – *незамещаемые* (nonreplaceable). Это означает, что их отображаемое агентом пользователя содержимое указано в коде. Например, элемент ****…**** – незамещаемый элемент, и агент пользователя (user agent) будет отображать текст. Это выполняется для абзацев, заголовков, ячеек таблиц, списков и почти всех остальных элементов HTML.

21.3 Схлопывание (сворачивание) вертикальных полей (*margin*)

Есть две ситуации в верстке, когда вертикальные поля схлопываются:

1) соседство двух элементов с вертикальными полями;
2) наличие вертикальных полей у родительского и дочернего элемента.

Внешне такой эффект выглядит так, будто поля накладываются друг на друга.

21.4 Строчные (*внутристрочные, inline*) *элементы*

Строчными называются элементы web-страницы, которые являются непосредственно частью строки. К строчным элементам относятся элементы **<img>**, ****, **<a>**, **<q>**, **<code>** и др. В основном они используются для изменения вида текста.

Характерные особенности строчных элементов:

– внутрь строчных элементов допустимо помещать текст или другие строчные элементы. Вставлять блочные элементы внутрь строчных ***запрещено***;

– эффект схлопывания полей не действует. Если рядом стоят два строчных элемента с определенными полями, то эти поля ***суммируются***;

– свойства, связанные с размерами (width, height) ***не применимы***. Их просто нет. На то она и строка;

– ширина элемента равна содержимому, плюс значения отступов, полей и границ;

– несколько строчных элементов идущих подряд располагаются на одной строке друг за другом и переносятся на следующую строку при необходимости;

– строчные элементы можно выравнивать по вертикали при помощи CSS-свойства **vertical-align**.

Контрольные вопросы

1 Какие существуют концепции ***визуального форматирования***?
2 В чем отличие ***замещаемого*** элемента от ***незамещаемого***?
3 Какие есть ***типы отображения*** элементов?
4 Что нужно сделать, чтобы ***не происходило*** схлопывание полей?
5 Как сделать элемент ***невидимым***?
6 Можно ли ***в строчные*** элементы вкладывать ***блочные***? И наоборот?

22 Лабораторная работа № 22. CSS. Позиционирование и свободное перемещение

Цель работы.

Изучение возможностей основных типов позиционирования в CSS. Научиться использовать позиционирование при создании web-документов.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить отчет в виде набора файлов.

22.1 Позиционирование в CSS

Сущность позиционирования состоит в том, чтобы любой элемент расположить в необходимом месте web-страницы. Позиционирование определяется свойством **position**, которое может иметь следующие значения:

position: static (значение по умолчанию для всех элементов в нормальном потоке)**;**

position: absolute;

position: fixed;

position: relative.

Указание точного месторасположения элементов осуществляется ***свойствами смещения***: **top**, **right**, **bottom**, **left**.

top – расстояние от верхнего края окна блока-контейнера (или браузера).

bottom – расстояние от нижнего края окна блока-контейнера (или браузера).

left – расстояние от левого края окна блока-контейнера (или браузера).

right – расстояние от правого края окна блока-контейнера (или браузера).

Свойства смещения работают со всеми позиционированными элементами, кроме имеющих значение **static**.

22.2 Абсолютное позиционирование

Абсолютное позиционирование задаётся при установке свойства элемента **«position: absolute»**.

При абсолютном позиционировании элемент:

– выводится из общего потока;

– становится независимым от других элементов;

– может накладываться на другие элементы.

При абсолютном позиционировании положение элемента задаётся только свойствами смещения этого элемента (**bottom**, **left**, **right**, **top**) относительно блок-контейнера.

Блок-контейнером считается ближайший предок (любой), значение свойства **position** которого отлично от **static**. Если таких предков нет, то блок-контейнером считается начальный блок-контейнер.

Важные моменты:

– абсолютно позиционированные элементы перемещаются вместе с документом при прокрутке;

– свойства **left** и **top** имеют приоритет выше, чем свойства **right** и **bottom**;

– элементы можно спрятать. Для этого можно задать отрицательным либо **left**, либо **top**. Элемент выйдет за границы окна браузера, полоса прокрутки при этом не возникнет;

– если задать **left** больше, чем ширина окна браузера либо отрицательное значение **right**, то возникнет горизонтальная полоса прокрутки. Аналогично будет и с **top**, только полоса прокрутки будет вертикальной.

22.3 Фиксированное позиционирование

Фиксированное позиционирование задаётся **position:fixed** и по своей сути очень похоже на абсолютное позиционирование. Элемент с фиксированным позиционированием также выводится из общего потока и не зависит от расположения других элементов. Положение элемента **не** изменяется при

прокрутке. Элемент, также как и при абсолютном позиционировании может накладываться на другие. Ещё одной особенностью фиксированного позиционирования является то, что при выходе содержимого за пределы видимой области **не** возникает полос прокрутки.

Фиксированное позиционирование используется для создания меню, вкладок, заголовков, т. е. таких элементов, которые всегда должны быть видны пользователю.

22.4 Относительное позиционирование

Относительно позиционирование задается свойством **position: relative**. При таком способе позиционирования положение элемента определяется относительно краёв элемента родителя и сам элемент не выводится из основного потока. Расстояние до краёв родительского элемента задаётся свойствами: **bottom**, **left**, **right**, **top**.

Важные моменты:

– данный тип позиционирования не применяется к элементам таблицы;

– если при смещении элемента образовалось пустое место, оно не занимается ниже или вышележащими элементами;

– относительно позиционированный элемент образует блок-контейнер для других элементов.

22.5 Липкое позиционирование

Элемент остается в нормальном потоке документа до тех пор, пока не пересечет определенное пороговое положение, после чего он рассматривается как фиксированный элемент (извлекается из нормального потока, но с сохранением места в нормальном потоке). Теперь он рассматривается как абсолютно позиционируемый элемент в своем содержащем блоке (блок-контейнере). Как только элемент пересекает пороговое положение в обратном направлении, он возвращается в исходное место в нормальном потоке документа.

22.6 Плавающие элементы

Плавающее поведение элемента осуществляется заданием свойства **float**. Оно применятся для создания красивого эффекта обтекания какого-либо элемента содержимым.

Возможные значения свойства **float**:

none – без обтекания;

left – выравнивание по левому краю, обтекание по правому краю;

right – выравнивание по правому краю, обтекание по левому краю.

22.7 Свойство z–индекс

Свойство z-индекс предназначено для работы с элементами страницы в трехмерном пространстве. Для этого вводится третья ось – ось Z. Так как

страницу мы видим как двумерную, элементы накладываются слой за слоем друг на друга. Управлять подобным наложением можно с помощью свойства **z-index**.

Возможные значения свойства **z-index**:

auto – элементы накладываются друг на друга в том порядке, в каком они были указаны в коде HTML;

целое число – чем больше число, тем более высокую позицию элемент занимает по оси Z и соответственно перекрывает все элементы, расположенные ниже по этой оси.

Контрольные вопросы

1 В чем сущность ***позиционирования***? Каким ***свойством*** оно определяется?
2 Какие свойства являются ***свойствами смещения***?
3 Какие ***базовые схемы размещения*** элементов есть в CSS?
4 Что такое ***нормальный поток***?
5 В чем отличие ***абсолютного позиционирования*** от ***относительного***?
6 Каким свойством задаются ***плавающие элементы***?

23 Лабораторная работа № 23. Основы Flexbox

Цель работы.

Изучение основ модели Flexbox для разработки web-документов.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.

2 Пройти учебную игру (результаты представить преподавателю при защите лабораторной работы).

3 Выполнить задание, полученное у преподавателя.

4 Оформить отчет в виде набора файлов.

Основные теоретические положения.

Flexbox – это общее название для модуля Flexible Box Layout, который входит в CSS3. Данный модуль определяет особый режим компоновки/верстки пользовательского интерфейса, который называется *flex layout* (*гибкий макет*). В этом плане Flexbox предоставляет иной подход к созданию пользовательского интерфейса, который отличается от табличной или блочной верстки.

Главной характеристикой `flex-контейнера` является способность менять ширину или высоту дочерних элементов, чтобы наиболее оптимально заполнить доступное пространство при разных размерах экрана (*адаптивная верстка*).

Основными составляющими компоновки *flexbox* являются: flex-контейнер (*flex container*) и flex-элементы (*flex items*).

Flex-контейнер – родительский элемент, в котором содержатся flex-элементы. Flex-контейнер определяется установкой свойства display в значение flex или inline-flex.

Flex-элемент – каждый дочерний элемент flex-контейнера становится flex-элементом. Текст, который напрямую содержится в flex-контейнере, оборачивается анонимным flex-элементом.

main axis (центральная ось) – это условная ось во flex-контейнере, вдоль которой позиционируются flex-элементы.

main start и *main end* – описывают начало и конец центральной оси, а расстояние между между ними обозначается как *main size*.

cross axis (поперечная ось) – она перпендикулярна основной оси.

cross start и *cross end* – начало и конец поперечной оси. Расстояние между ними описывается термином *cross size*.

Контрольные вопросы

1 Как создать flex-контейнер?
2 Как создать flex-элемент?
3 Назовите оси flex-контейнера?
4 Какое свойство задаёт направление основных осей в контейнере?
5 Как разрешить перенос flex-элементов внутри flex-контейнера?

24 Лабораторная работа № 24. Применение фреймворка Bootstrap

Цель работы.

Изучение порядка создания и использования системы разметки Bootstrap при разработке адаптивных web-страниц.

Порядок выполнения работы.

1 Ознакомиться с основными теоретическими положениями.
2 Выполнить задание, полученное у преподавателя.
3 Оформить отчет в виде набора файлов.

Основные теоретические положения.

Bootstrap – набор инструментов для создания адаптивных сайтов и web-приложений.

Для начала необходимо скачать Bootstrap с сайта разработчика.

Полная версия содержит все необходимые скрипты и стили. Внутри находятся три папки: css, fonts и js.

В папках со стилями и скриптами предложено две версии файлов – исходная и компактная (в имени содержится min).

Компактная отличается лишь размером файла и снижением читаемости кода. Лучше всего на рабочий сайт добавлять именно эту версию, т. к. это немного ускорит загрузку web-страниц.

Подключение стилей и скриптов bootstrap происходит в html-файле.

Если в процессе работы потребуется переопределить стили каких-то элементов, то можно подключить ещё один собственный стилевой файл и в нём задавать необходимые свойства.

В Bootstrap используются некоторые элементы HTML и свойства CSS, которые требуют использования HTML5 Doctype обязательно.

24.1 Сетка

Особенность работы с Bootstrap заключается в использование сетки (grid), в основе которой лежит 12-колоночный макет.

Сами колонки не обязательно должны иметь одинаковую ширину, можно комбинировать любое число колонок, главное, чтобы они в сумме давали 12.

24.2 Создание колонок

Так, чтобы создать макет из четырех колонок используем ** ** с классом **container**, внутри которого будут располагаться колонки. Сами колонки находятся внутри ** ** с классом **row** и содержат имена вида **col-md-N**, где N – число колонок от 1 до 12.

Класс **container** создаёт макет фиксированной ширины, значение которой зависит от размера устройства. Для мониторов максимальная ширина составляет 1140 пикселей, для смартфонов макет будет занимать всю доступную ширину. Если не требуется ограничивать ширину макета, то вместо класса **container** следует использовать **container-fluid**.

Для добавления пустого пространства между колонками предназначен класс **offset-md-N**, где N изменяется от 0 до 12. Отступ добавляется слева от текущей колонки.

Однако стоит понимать, что отступы добавляются к общему числу колонок, сумма которых не должна превышать 12, в противном случае колонки начнут перемещаться на другую строку.

Можно организовать порядок колонок, опять же это делается с помощью класса **order-md-N**. Здесь N может меняться от 0 до 12.

При вёрстке сложных макетов двенадцати колонок может не хватить, к тому же в одной колонке могут встречаться ещё дополнительные. Для этого потребуются вложения одних колонок в другие.

Чтобы создать вложенные колонки опять добавляем ** ** с классом **row**, который содержит желаемую структуру вложенных колонок. Таким образом, можно сверстать какие угодно сложные макеты.

24.3 Адаптивный дизайн

Адаптивным дизайном называется способ вёрстки, когда ширина макета веб-страницы подстраивается под ширину устройства.

В Bootstrap уже заложены возможности адаптивного дизайна. Если сделать простой многоколоночный макет, а затем начать уменьшать ширину окна браузера, то можно заметить, что изменяется и сам макет. Из этого следует, что на разных устройствах один и тот же макет будет выглядеть по-разному.

Существуют классы, с помощью которых можно менять расположение блоков на разных устройствах.

Контрольные вопросы

1 Для чего предназначен Bootstrap?
2 Что такое адаптивная вёрстка?
3 В чём особенность сетки Bootstrap?
4 Какие классы разметки в Bootstrap?
5 Как создать вложенные колонки в макете?

Список литературы

1 **Бройдо, В. Л.** Вычислительные системы, сети и телекоммуникации: учебник / В. Л. Бройдо. – 4-е изд. – Санкт-Петербург: Питер, 2013. – 560 с. : ил.

2 **Станек, У.** Windows 7 для продвинутых. Настройка, работа и администрирование / У. Станек. – Санкт-Петербург: Питер, 2011. – 576 с: ил.

3 **Гуриков, С. Р.** Введение в программирование на языке Visual Basic for Applications (VBA): учебное пособие / С. Р. Гуриков. – Москва: ИНФРА-М, 2020. – 317 с.

4 **Гуриков, С. Р.** Интернет-технологии: учебное пособие / С. Р. Гуриков. – Москва: ФОРУМ; ИНФРА-М, 2017. – 184 с.

5 **Коваленко, В. В.** Проектирование информационных систем: учебное пособие / В. В. Коваленко. – 2-е изд., перераб. и доп. – Москва: ИНФРА-М, 2021. – 357 с.

6 **Фрейн, Бен.** HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Бен Фрейн. – Санкт-Петербург: Питер, 2014. – 304 с: ил.

7 **Гоше, Х. Д.** HTML5. Для профессионалов: пер. с англ. / Х. Д. Гоше. – 2-е изд. – Санкт-Петербург: Питер, 2015. – 560 с.: ил.

8 **Дронов, В. А.** HTML 5, CSS 3 и Web 2.0. Разработка современных web-сайтов / В. А. Дронов. – Санкт-Петербург: БХВ-Петербург, 2011. – 414 с.

9 **Дакетт, Джон.** HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. – 2-е изд. – Санкт-Петербург: Эксмо, 2017. – 923 с.

10 **Дунаев, В. В.** HTML, скрипты и стили / В. В. Дунаев. – Санкт-Петербург: БХВ-Петербург, 2011. – 810 с.

11 **Евсеев, Д. А.** Web-дизайн в примерах и задачах: учебное пособие / Д. А. Евсеев, В. В. Трофимов; под ред. В.В. Трофимова. – Москва: КНОРУС, 2010. – 272 с.

12 **Макфарланд, Д.** Большая книга CSS / Д. Макфарланд. – 2-е изд. – Санкт-Петербург: Питер, 2012. – 560 с.: ил.

13 **Мак-Дональд, М.** HTML5. Недостающее руководство: пер. с англ / М. Мак-Дональд. – Санкт-Петербург: БХВ-Петербург, 2012. – 480 с. : ил.

14 **Матросов, А. В.** HTML 4.0 / А. В. Матросов, А. О. Сергеев, М. П. Чаунин. – Санкт-Петербург: БХВ-Петербург, 2007. – 672 с.: ил.

15 **Мейер, Э.** CSS-каскадные таблицы стилей: пер. с англ. / Э. Мейер. – 3-е изд. – Санкт-Петербург: Символ-Плюс, 2010. – 576 с.: ил.

16 **Комолова, Н.** HTML, XHTML и CSS / Н. Комолова, Е. Яковлева. – Санкт-Петербург: Питер, 2012. – 304 с.: ил.

17 **Немцова, Т. И.** Компьютерная графика и web-дизайн: учебное пособие / Т. И. Немцова, Т. В. Казанкова, А. В. Шнякин. – Москва: ФОРУМ; ИНФРА-М, 2014. – 400 с.