

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

# **МЕТОДИЧЕСКИЕ УКАЗАНИЯ**

по выполнению лабораторных работ  
по дисциплине «Современные технологии  
программирования»  
для направления подготовки 09.03.02  
«Информационные системы и технологии»

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	5
Тема: Введение в web-технологии: структура и принципы Web ЛАБОРАТОРНАЯ РАБОТА 1. HTML 5. Работа с Web-формами.....	10
Тема: Введение в web-технологии: структура и принципы Web ЛАБОРАТОРНАЯ РАБОТА 2. HTML 5. Drag and Drop. Оформление страницы в HTML 5. Мультимедиа.....	34
Тема: Сценарии и приложения, выполняющиеся на стороне клиента ЛАБОРАТОРНАЯ РАБОТА 3. HTML 5. Работа с графикой. Canvas. Анимация.....	49
Тема: Введение в C# и платформу Visual Studio.Net ЛАБОРАТОРНАЯ РАБОТА 4. Язык C#. Структура программы на языке C#...	57
Тема: Введение в C# и платформу Visual Studio.Net ЛАБОРАТОРНАЯ РАБОТА 5. Работа с массивами и строками.....	67
Тема: Архитектура web-приложений ASP.NET. Разработка web-приложений на платформе .NET. ЛАБОРАТОРНАЯ РАБОТА 6. Разработка Web-приложений с помощью ASP.Net.....	83
Тема: Клиент-серверные технологии Web. Протокол HTTP ЛАБОРАТОРНАЯ РАБОТА 7. Серверные элементы управления ASP.Net...	89
Тема: Архитектура web-приложений ASP.NET. Разработка web-приложений на платформе .NET. ЛАБОРАТОРНАЯ РАБОТА 8. Работа с источниками данных в ASP.Net.....	98
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА И ИНТЕРНЕТ-РЕСУРСЫ.....	114

## ВВЕДЕНИЕ

*Целью* изучения дисциплины «Современные технологии программирования» является формирование навыков по разработке, документированию и сопровождению сетевых приложений, расширение профессионального кругозора студентов, повышение программистской культуры. формирование набора общекультурных, общепрофессиональных и профессиональных компетенций будущего магистра по направлению подготовки 09.03.02 «Информационные системы и технологии».

*Задачами* изучения дисциплины «Современные технологии программирования» являются

- 1) формирование представлений об основных принципах функционирования Internet-приложений и навыков их разработки,
- 2) изучение основных подходов, платформ, технологий и инструментов проектирования Internet-приложений;
- 3) формирование набора профессиональных компетенций.

# Тема: Введение в web-технологии: структура и принципы Web

## ЛАБОРАТОРНАЯ РАБОТА 1

### HTML5. РАБОТА С WEB-ФОРМАМИ

**Цель работы:** познакомить студентов с базовыми приемами создания Web-страниц средствами HTML, CSS и JavaScript, инструментальным средством WebMatrix; основами работы с Web-формами: создание Web-форм; стилизация форм; валидация вводимых значений; задание маски ввода.

#### Формируемые компетенции или их части:

1. ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий

#### Теоретическая часть

Алгоритм установки WebMatrix может быть представлен следующей последовательностью действий:

1. Загрузите установщик по ссылке

<http://www.microsoft.com/web/gallery/install.aspx?appid=webmatrix>.

2. Запустите установщик.

3. Ознакомьтесь с параметрами установки и устанавливаемыми компонентами.

4. Для дальнейшей работы создайте сайт на основе шаблона (рисунок 1.1), а именно на основе шаблона "Пустой сайт" (рисунок – 1.2).



Рисунок 1.1 – Создание сайта на основе шаблона

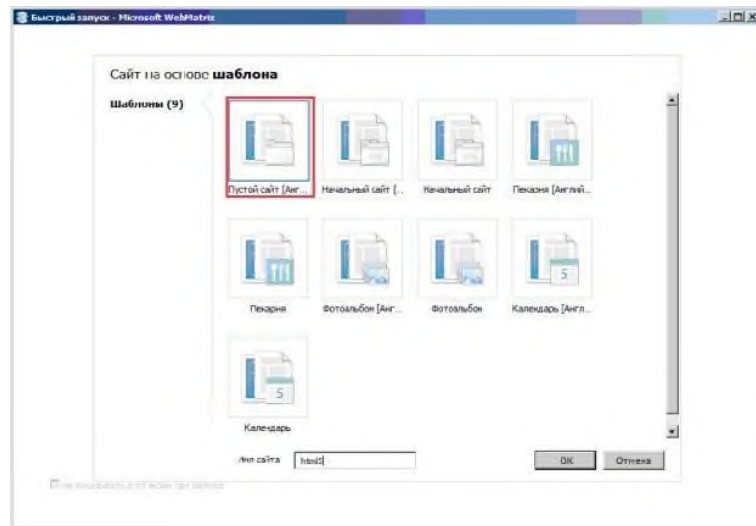


Рисунок 1.2 – Создание сайта на основе пустого шаблона

5. После окончания процесса создания проекта сайта, перейдите к разделу **Файлы** (рисунок 1.3) и создайте папку для создаваемого сайта (рисунок 1.4).

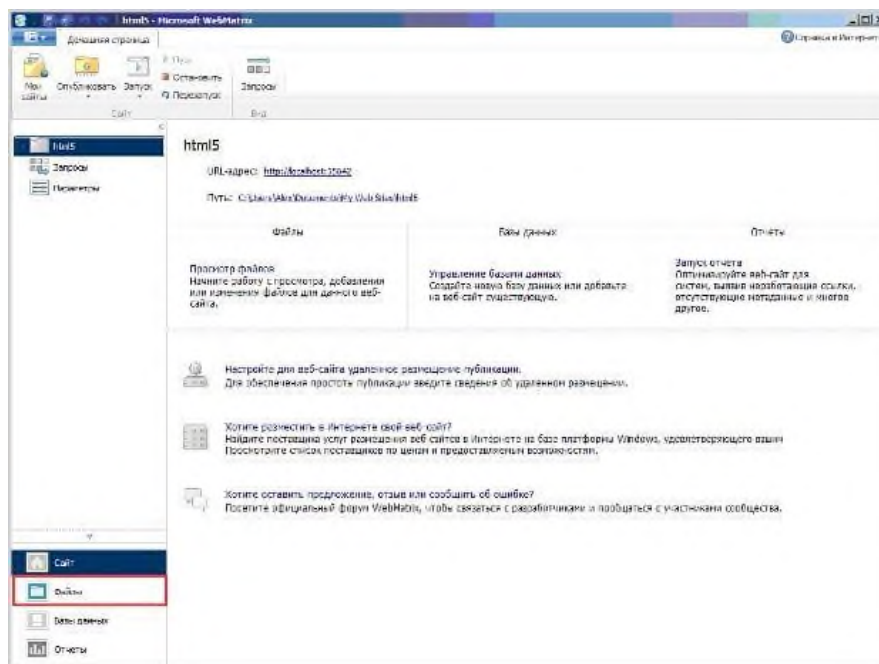


Рисунок 1.3 – Переход к просмотру и изменению файлов сайта

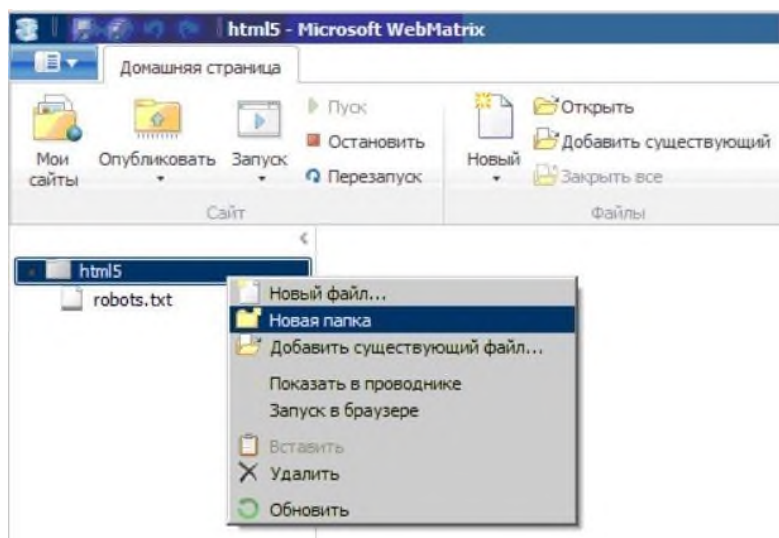


Рисунок 1.4 – Создание новой папки

### *Создание Web- страницы с подгружаемым содержимым*

В качестве примера рассмотрим создание страницы с подгружаемым содержимым, например, теоретический материал к одной из лекций курса

Для удобства необходимо создать еще несколько папок, которые будут использоваться для хранения стилей, Web- сценариев и, собственно, для тех страниц, содержимое которых будем подгружать (рисунок 1.5).

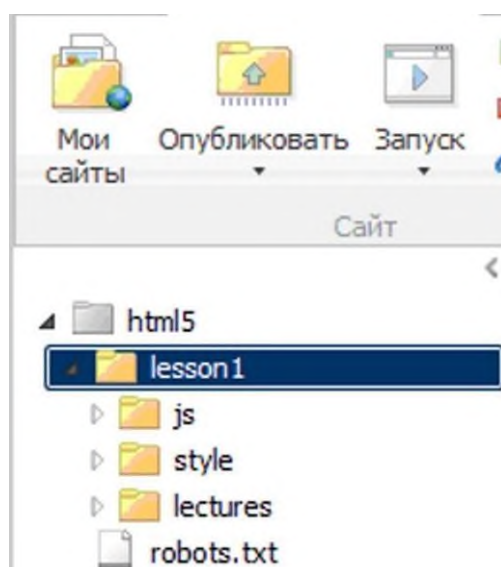


Рисунок 1.5 – Иерархия каталогов

### *Создание страниц - содержимого*

Для начала создадим несколько Web-страниц, содержимое которых, затем будем погружать в основную Web - страницу.

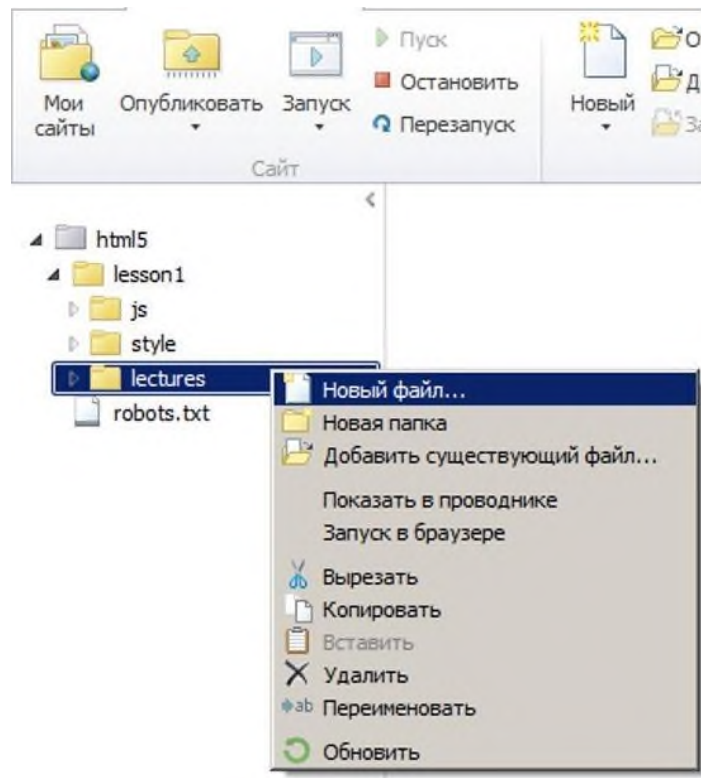


Рисунок 1.6 – Добавление нового файла в папку

В открывшемся окне, выберем тип создаваемого файла (рисунок 1.7)

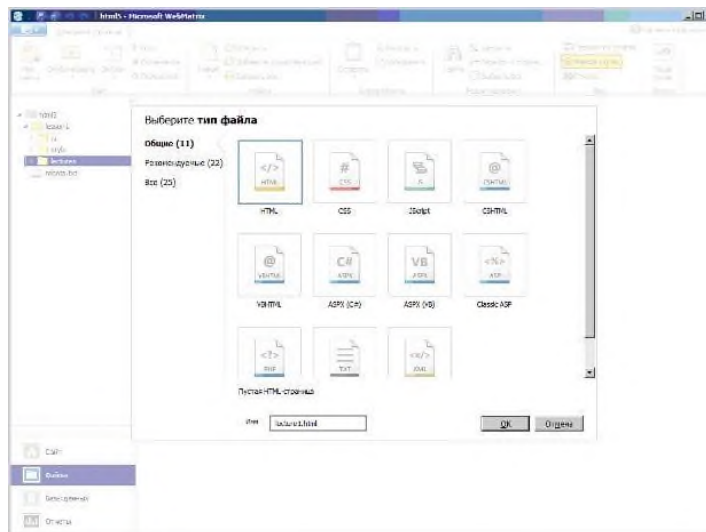


Рисунок 1.7 – Выбор типа создаваемого файла

### *Создание стилей содержимого*

Для управления содержимым создадим CSS - файл:

```
h1, h2 { text-align:center; }
```

```
.text { text-align:justify }
```

```
.annotation { font-style:italic; }
```

```
table.normal { border: 1px solid black; border-right: none; border-bottom:
```

```
none }
```

```
table.normal tr td { border-bottom: 1px solid black; border-right: 1px solid
                    black }
```

```
table.normal thead { text-align:center; font-weight:bold } table.normal
```

```
caption { text-align:right }
```

```
dt.terms { font-style:italic; font-weight:bold }
```

### *Создание основной страницы*

Создадим основную *HTML* страницу для выполнения задания. Для этого, во-первых, на страницу необходимо добавить четыре контейнера, во-вторых, создать *css-файл* для управления размещением контейнеров.

С добавлением контейнеров на страницу все просто:

```
<div id="header">HEADER</div>
```

Также не должно быть сложностей и с их позиционированием относительно друг друга:

```
#top { height: 15%; width:100%; }
```

### *Реализация подгрузки содержимого*

Осталось немного, а именно: создать полосу навигации и организовать подгрузку содержимого в блок *content*.

Полоса навигации, к примеру может быть создана следующим образом:

```
<div id="navigation"><a href="javascript:/" on-
click="loadContent('lectures/lecture1.html')"> Лекция № 1</a></div>
```

Теперь необходимо поместить элемент *iframe* в контейнер *content*:

```
<div id="content"><iframe id="target" src=""></iframe></div>
```

И, наконец, создать *JavaScript* - файл, в котором разместим функцию, обрабатывающую клик по ссылке:

```
function loadContent(path)
{
  document.getElementById('target').src = path;
}
```

Для завершения задания необходимо:

- создать несколько страниц – источников;
- оформить панель навигации;

- создать стили для оформления содержимого контейнеров.

### *Создание Web-форм*

По аналогии с предыдущим практическим заданием создадим папку для стилей и основную *HTML* - страницу.

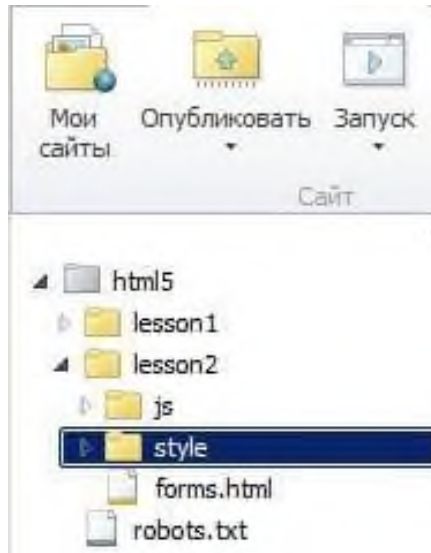


Рисунок – 1.8 – Иерархия каталогов

Рассмотрим по шагам создание требуемой страницы с регистрационной формой.

Шаг 1. Создание формы регистрации

```
<form id="registration">
  <fieldset>
    <legend>Форма регистрации</legend>
  </fieldset>
</form>
```

Тег `<fieldset>` используется для логической группировки объектов формы. Тег `<legend>` определяет заголовок.

Шаг 2. Добавление поля для ввода ФИО пользователя:

```
<fieldset>
  <legend>Форма регистрации</legend>
  <label for=name>ФИО</label>
  <input id=name name=name type=text>
</fieldset>
```

На данном шаге форма будет выглядеть в браузере следующим образом:

Рисунок 1.9 – Промежуточный результат на шаге 2

Шаг 3. Создание поля для ввода *@-mail* и номера телефона, почтового индекса и города:

```
<fieldset>
  <legend>Форма регистрации</legend>
  <label for=name>ФИО</label>
  <input id=name name=name type=text>
  <label for=email>Email</label>
  <input id=email name=email type=email>
  <label for=phone>Номер телефона</label>
  <input id=phone name=phone type=tel>
</fieldset>
```

Рисунок 1.10 – Промежуточный результат на шаге 2

Шаг 4. Для выравнивания элементов формы построчно и относительно друг друга необходимо создать таблицу с двумя колонками и без видимых границ внесем изменения в код следующим образом:


```
<fieldset>
  <legend>Форма регистрации</legend>
  <table class="alignment">
    <tr>
      <td> <label for=name>ФИО</label> </td>
      <td> <input id=name name=name type=text> </td>
    </tr>
    <tr>
      <td>
```

```

<td> <label for=email>Email</label> </td>
<td> <input id=email name=email type=email> </td>
</tr>
<tr>
<td> <label for=phone>Номер телефона</label> </td>
<td> <input id=phone name=phone type=tel> </td>
</tr>
</table>
</fieldset>

```

В результате получим следующее:



The image shows a web form titled "Форма регистрации" (Registration Form). It contains three input fields arranged vertically. The first field is labeled "ФИО" (Full Name), the second is labeled "Email", and the third is labeled "Номер телефона" (Phone Number). Each label is positioned to the left of its corresponding input box.

Рисунок 1.11 – Промежуточный результат на шаге 4

Шаг 5. Добавление поля для ввода даты рождения с помощью `type = date` (работает не со всеми браузерами). Для этого к таблице добавим следующую строку:

```

<tr>
<td><label for=dateofbirth>Дата рождения</label></td>
<td><input id=dateofbirth name=dateofbirth type="date" /></td>
</tr>

```

В случае, если браузер поддерживает данный тип элементов управления, то получим следующее:

Форма регистрации

ФИО

Email

Номер телефона

Город

Дата рождения Day.Month.Year ▾

<< < Май 2002 ▾ > >>

Пн	Вт	Ср	Чт	Пт	Сб	Вс
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Today Clear

Рисунок 1.12 – Промежуточный результат на шаге 5

Шаг 6. Добавление элемента *fieldset* для логической группировки, объединяющий элементы, указывающие адрес и почтовый индекс:

```

<fieldset>
  <legend>Параметры доставки</legend>
  <table>
    <tr>
      <td><label for=address>Адрес</label></td>
      <td><textarea id=address name=address rows=5 ></textarea></td>
    </tr>
    <tr>
      <td><label for=postcode>Почтовый индекс</label></td>
      <td><input id=postcode name=postcode type=text ></td>
    </tr>
  </table>
</fieldset>

```

Поскольку написание полного адреса, скорее всего займет не одну строку, зададим атрибут `rows`, со значением равным 5 для соответствующего

тега.

Форма регистрации	
ФИО	<input type="text"/>
Email	<input type="text"/>
Номер телефона	<input type="text"/>
Город	<input type="text"/>
Дата рождения	<input type="text"/>
Параметры доставки	
Адрес	<input type="text"/>
Почтовый индекс	<input type="text"/>

Рисунок 1.13 – Промежуточный результат на шаге 6

Шаг 7. Выделение в отдельную логическую группу элементов управления, с помощью которого пользователь укажет предпочитаемый способ получения уведомлений:

```

<fieldset>
<legend>Предпочитаемый способ получения уведомлений</legend>
<table>
<tr>
<td><input id=emailmessage name=emailmessage type=radio></td>
<td><label for=emailmessage>По Email</label></td>
</tr>
<tr>
<td><input id=phonemessage name=phonemessage type=radio></td>
<td><label for=phonemessage>По телефону</label></td>
</tr>
<tr>
<td><input id=nomessagee name=nomessage type=radio></td>
<td><label for=nomessage>Не уведомлять меня</label></td>

```

```

</tr>
</table>
</fieldset>

```

Форма регистрации

ФИО

Email

Номер телефона

Город

Дата рождения

---

Параметры доставки

Адрес

Почтовый индекс

---

Предпочитаемый способ получения уведомлений:

По Email

По телефону

Не уведомлять меня

Рисунок 1.14 – Промежуточный результат на шаге 7

Шаг 8. Добавление на страницу кнопки

```

<fieldset>
  <button type=submit>Отправить данные</button>
</fieldset>

```

Форма регистрации	
ФИО	<input type="text"/>
Email	<input type="text"/>
Номер телефона	<input type="text"/>
Город	<input type="text"/>
Дата рождения	<input type="text"/>
Параметры доставки	
Адрес	<input type="text"/>
Почтовый индекс	<input type="text"/>
Предпочитаемый способ получения уведомлений	
<input type="radio"/> По Email <input type="radio"/> По телефону <input type="radio"/> Не уведомлять меня	
<input type="button" value="Отправить данные"/>	

Рисунок 1.15 – Промежуточный результат на шаге 8

### Стилизация форм

Шаг 1. Создание стиля для всей формы

```
form#registration {
  background: #1E90FF;
  - moz-border-radius: 5px;
  - webkit-border-radius: 5px;
  - khtml-border-radius: 5px;
  border-radius: 5px;
  counter-reset: fieldsets;
  padding: 20px;
  width: 400px;
```

Шаг 2. Удаление границы у *fieldset* и добавление отступа

```
form#registration fieldset {
```

```
border: none;
margin-bottom: 10px;
```

Шаг 3. Стилизация legends:

```
form#registration legend {
color: #384313;
font-size: 16px;
font-weight: bold;
padding-bottom: 10px;
text-shadow: 0 1px 1px #C1F7FF;
```

Шаг 4. Стилизация элементы label, input и button. Все label должны выглядеть одинаково, кроме поля label, которое используется для элементов radio. Выравниваем их по левому краю и придадим ширину:

```
form#registration label {
float: left;
font-size: 13px;
width: 110px;
}
form#registration fieldset fieldset label {
background:none no-repeat left 50%;
line-height: 20px;
padding: 0 0 0 30px;
width: auto;
}
form#registration button {
background: #87CEEB;
border: none;
- moz-border-radius: 20px;
- webkit-border-radius: 20px;
- khtml-border-radius: 20px;
border-radius: 20px;
color: #ffffff;
```

```
display: block;
font: 18px Georgia, "Times New Roman", Times, serif;
letter-spacing: 1px;
margin: auto;
padding: 7px 25px;
text-shadow: 0 1px 1px #C1F7FF;
text-transform: uppercase;
```

### *Валидация вводимых значений*

Рассмотрим следующий участок *HTML* кода:

```
<input id=email name=email type=email />
```

Атрибут типа равен «@-mail», а не «text». Самое ценное в новых HTML-типах input в том, что вы можете использовать их сейчас и они будут работать на том или ином уровне в любом браузере. Когда браузер встречает один из этих типов, происходит одно из двух.

Если браузер не поддерживает новые input-типы, объявление типа не распознается. В таком случае браузер корректно сокращает функциональность и интерпретирует элемент как type="text" .

Чтобы сделать элемент управления обязательным к заполнению, достаточно вставить в создающий его тег атрибут REQUIRED. Это атрибут тега без значения.

```
<input id=name name=name type=text required>
```

Задать минимальное и максимальное значение и шаг числовых значений можно только для полей ввода числовых величин. Другие элементы управления, в том числе и обычные поля вида, эту возможность не поддерживают.

Для задания минимального значения числа используется атрибут тега MIN, а для задания максимального значения – атрибут тега MAX. В качестве их значений указываются числа.

```
<INPUT TYPE=" number" ID=" txtAge" REQUIRED MIN=" 1" MAX=" 100" >
```

### *Задание маски ввода*

Маска ввода задаёт формат, которому должно соответствовать вводимое значение. Как правило, она указывается для обычных полей ввода.

Для указания маски используется атрибут стиля PATTERN. В качестве его значения указывается регулярное выражение, собственно, и задающее маску ввода.

```
<input id=phone name=phone type=tel pattern=" (d{3,4}) d{2,3}-d{2}-d{2}" >
```

Эта Web-форма содержит поле ввода, в котором указывается номер телефона в формате ([x]xxx) [x]xx-xx-xx, где x- цифра.

Если посетитель введёт в поле ввода с указанной маской значение, не соответствующее данной маске, то увидит всплывающее сообщение с предупреждением о неверном формате введенного значения.

Для окончания формирования формы необходимо:

- поля ФИО, @-mail и номер телефона, определить, как обязательные к заполнению;
- создать маску ввода для поля «Номер телефона»;
- осуществить проверку вводимых значений поля «Почтовый индекс»;
- ознакомиться с материалами для самостоятельного изучения.

### *Создание Web-форм*

Рассмотрим по шагам создание требуемой страницы с регистрационной формой.

Шаг 1. Создание формы регистрации

```
<form id="registration">
  <fieldset>
    <legend>Форма регистрации</legend>
  </fieldset>
</form>
```

Тег <fieldset> используется для логической группировки объектов формы. Тег <legend> определяет заголовок.

Шаг 2. Добавление поля для ввода ФИО пользователя:

```

<fieldset>
  <legend>Форма регистрации</legend>
  <label for=name>ФИО</label>
  <input id=name name=name type=text>
</fieldset>

```

Шаг 3. Создание поля для ввода @-mail и номера телефона, почтового индекса и города:

```

<fieldset>
  <legend>Форма регистрации</legend>
  <label for=name>ФИО</label>
  <input id=name name=name type=text>
  <label for=email>Email</label>
  <input id=email name=email type=email>
  <label for=phone>Номер телефона</label>
  <input id=phone name=phone type=tel>
</fieldset>

```

Шаг 4. Для выравнивания элементов формы построчно и относительно друг друга необходимо создать таблицу с двумя колонками и без видимых границ внесем изменения в код следующим образом:

```

<fieldset>
  <legend>Форма регистрации</legend>
  <table class="alignment">
    <tr>
      <td> <label for=name>ФИО</label> </td>
      <td> <input id=name name=name type=text> </td>
    </tr>
    <tr>
      <td> <label for=email>Email</label> </td>
      <td> <input id=email name=email type=email> </td>
    </tr>
  </table>

```

```

<td> <label for=phone>Номер телефона</label> </td>
<td> <input id=phone name=phone type=tel> </td>
</tr>
</table>
</fieldset>

```

Шаг 5. Добавление поля для ввода даты рождения с помощью `type = date` (работает не со всеми браузерами). Для этого к таблице добавим следующую строку:

```

<tr>
<td><label for=dateofbirth>Дата рождения</label></td>
<td><input id=dateofbirth name=dateofbirth type="date" /></td>
</tr>

```

Шаг 6. Добавление элемента *fieldset* для логической группировки, объединяющий элементы, указывающие адрес и почтовый индекс:

```

<fieldset>
<legend>Параметры доставки</legend>
<table>
<tr>
<td><label for=address>Адрес</label></td>
<td><textarea id=address name=address rows=5 ></textarea></td>
</tr>
<tr>
<td><label for=postcode>Почтовый индекс</label></td>
<td><input id=postcode name=postcode type=text ></td>
</tr>
</table>
</fieldset>

```

Поскольку написание полного адреса, скорее всего займет не одну строку, зададим атрибут `rows`, со значением равным 5 для соответствующего тега.

Шаг 7. Выделение в отдельную логическую группу элементов управле-

ния, с помощью которого пользователь укажет предпочитаемый способ получения уведомлений:

```
<fieldset>
<legend>Предпочитаемый способ получения уведомлений</legend>
<table>
  <tr>
    <td><input id=emailmessage name=emailmessage type=radio></td>
    <td><label for=emailmessage>По Email</label></td>
  </tr>
  <tr>
    <td><input id=phonemessage name=phonemessage type=radio></td>
    <td><label for=phonemessage>По телефону</label></td>
  </tr>
  <tr>
    <td><input id=nomessagee name=nomessage type=radio></td>
    <td><label for=nomessage>Не уведомлять меня</label></td>
  </tr>
</table>
</fieldset>
```

Шаг 8. Добавление на страницу кнопки

```
<fieldset>
  <button type=submit>Отправить данные</button>
</fieldset>
```

### *Стилизация форм*

Шаг 1. Создание стиля для всей формы

```
form#registration {
  background: #1E90FF;
  - moz-border-radius: 5px;
  - webkit-border-radius: 5px;
  - khtml-border-radius: 5px;
```

```
border-radius: 5px;
counter-reset: fieldsets;
padding: 20px;
width: 400px;
```

Шаг 2. Удаление границы у *fieldset* и добавление отступа

```
form#registration fieldset {
border: none;
margin-bottom: 10px;
```

Шаг 3. Стилизация legends:

```
form#registration legend {
color: #384313;
font-size: 16px;
font-weight: bold;
padding-bottom: 10px;
text-shadow: 0 1px 1px #C1F7FF;
```

Шаг 4. Стилизация элементы label, input и button. Все label должны выглядеть одинаково, кроме поля label, которое используется для элементов radio. Выравниваем их по левому краю и придадим ширину:

```
form#registration label {
float: left;
font-size: 13px;
width: 110px;
}
form#registration fieldset fieldset label {
background:none no-repeat left 50%;
line-height: 20px;
padding: 0 0 0 30px;
width: auto;
}
form#registration button {
```

```

background: #87CEEB;
border: none;
- moz-border-radius: 20px;
- webkit-border-radius: 20px;
- khtml-border-radius: 20px;
border-radius: 20px;
color: #ffffff;
display: block;
font: 18px Georgia, "Times New Roman", Times, serif;
letter-spacing: 1px;
margin: auto;
padding: 7px 25px;
text-shadow: 0 1px 1px #C1F7FF;
text-transform: uppercase;

```

### *Валидация вводимых значений*

Рассмотрим следующий участок *HTML* кода:

```
<input id=email name=email type=email />
```

Атрибут типа равен «@-mail», а не «text». Самое ценное в новых HTML-типах input в том, что их можно использовать сейчас и они будут работать на том или ином уровне в любом браузере. Когда браузер встречает один из этих типов, происходит одно из двух.

Если браузер не поддерживает новые input-типы, объявление типа не распознается. В таком случае браузер корректно сокращает функциональность и интерпретирует элемент как type = "text" .

Чтобы сделать элемент управления обязательным к заполнению, достаточно вставить в создающий его тег атрибут REQUIRED. Это атрибут тега без значения.

```
<input id=name name=name type=text required>
```

Задать минимальное и максимальное значение и шаг числовых значений можно только для полей ввода числовых величин. Другие элементы

управления, в том числе и обычные поля вида, эту возможность не поддерживают.

Для задания минимального значения числа используется атрибут тега MIN, а для задания максимального значения – атрибут тега MAX. В качестве их значений указываются числа.

```
<INPUT TYPE=" number" ID=" txtAge" REQUIRED MIN=" 1" MAX=" 100" >
```

#### *Задание маски ввода*

Маска ввода задаёт формат, которому должно соответствовать вводимое значение. Как правило, она указывается для обычных полей ввода.

Для указания маски используется атрибут стиля PATTERN. В качестве его значения указывается регулярное выражение, собственно, и задающее маску ввода.

```
<input id=phone name=phone type=tel pattern=" (d{3,4}) d{2,3}-d{2}-d{2}" >
```

Эта Web-форма содержит поле ввода, в котором указывается номер телефона в формате ([x]xxx) [x]xx-xx-xx, где x- цифра.

Если посетитель введёт в поле ввода с указанной маской значение, не соответствующее данной маске, то увидит всплывающее сообщение с предупреждением о неверном формате введенного значения.

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт; устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить

об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### Задания

**Задание 1.** Подготовить рабочее место для выполнения практических занятий.

**Задание 2.** Создать Web-страницу на основе контейнерного дизайна.

**Задание 3.** Создайте страницу регистрации с Web-формами, по образцу многочисленных регистрационных Web-страниц, вида, представленного на рисунке.

Необходимо также создать соответствующие стили для оформления Web-формы и валидацию вводимых значений.

Для окончания формирования формы необходимо:

- поля ФИО, @-mail и номер телефона, определить, как обязательные к заполнению;
- создать маску ввода для поля «Номер телефона»;
- осуществить проверку вводимых значений поля «Почтовый индекс».

<i>ФИО</i>	<input type="text"/>
<i>Email</i>	<input type="text"/>
<i>Телефон</i>	<input type="text"/>
<i>Почтовый индекс</i>	<input type="text"/>
<i>Адрес</i>	<input type="text"/>
<i>Дата</i>	<input type="text"/>
<i>Предпочитаемый способ получения уведомлений</i>	<input type="radio"/> <i>По телефону</i> <input type="radio"/> <i>email</i>
<input type="button" value="Отправить данные"/>	

### **Указания по порядку выполнения работы**

При выполнении заданий к лабораторной работе использовать теоретический материал и практические примеры, приведенные в нем.

#### **Содержание отчета**

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;
- 3) формулировки индивидуальных заданий и порядка их выполнения.

#### **Контрольные вопросы**

1. Каким образом можно установить WebMatrix?
2. Дайте характеристику алгоритмам создания «страниц с подгружаемым содержимым», «страниц – содержимого».
3. Охарактеризуйте процесс создания стилей содержимого.
4. Алгоритм создания основной страницы.
5. Каким образом можно создать Web-форму в HTML 5?
6. Охарактеризуйте процесс стилизации (оформления) форм.
7. С какой целью используется валидация вводимых значений?
8. Каким образом можно задать маску ввода в Web-форме?

**Тема: Введение в web-технологии: структура и принципы Web**

### **ЛАБОРАТОРНАЯ РАБОТА 2**

#### **HTML 5. DRAG AND DROP. ОФОРМЛЕНИЕ СТРАНИЦЫ В**

#### **HTML 5. МУЛЬТИМЕДИА**

**Цель работы:** является формирования навыков работы с методами и элементами разметки, позволяющими менять местоположение любых элементов страницы при помощи мыши, на основе событийного механизма и *JavaScript API*, определенных спецификацией *HTML 5*; сформировать базовые навыки работы с элементами `<audio>` и `<video>`.

#### **Формируемые компетенции или их части:**

2. ОПК-6 Способен разрабатывать алгоритмы и программы,

пригодные для практического применения в области информационных систем и технологий

### Теоретическая часть

Создадим два контейнера, в один из которых (источник) поместим три элемента, элементы можно будет свободно перемещать из контейнера – источника, в целевой контейнер и обратно.

Для этого необходимо:

- создать контейнеры и элементы для перемещения;
- стилизовать элементы Web-страницы;
- создать функции для обработки следующих событий: перетаскивание объекта внутрь границ элемента; прохождение курсора "над" элементом во время осуществления операции перемещения; "освобождение" перетаскиваемого элемента в пределах элемента – цели; начало операции перемещения; окончание операции перетаскивания.

#### *Создание элементов страницы для перемещения*

Для того, чтобы можно было перемещать элемент в рамках страницы, достаточно просто добавить атрибут `draggable = true`. Создадим контейнеры для размещения элементов и элементы для последующего перемещения:

```
<div id="source" class="container" >
  <div id="firstDragElement" class="element" draggable="true">Text
1</div>
  <div id="secondDragElement" class="element" draggable="true">Text
2</div>
  <div id="thirdDragElement" class="element" draggable="true">Text
3</div>
</div>
<div id="target" class="container"></div>
```

#### *Стилизация элементов*

Для простоты восприятия примера, создадим следующие стили для элементов:

```

.element {
border: 2px solid black;
height: 50px;
width: 50px;
margin-left:20px;
margin-bottom: 10px;
text-align:center;
}

.container {
border: 2px solid red;
height: 200px;
width: 100px;
float:left;
margin: 50px;

```

### *Создание JavaScript-функции*

Отслеживать процесс *Drag and drop* позволяют следующие события: `dragstart`; `drag`; `dragenter`; `dragleave`; `dragover`; `drop`; `dragend`.

События `dragenter`, `dragover` и `dragleave` можно использовать для того, чтобы сделать процесс переноса более наглядным.

Для обработки процесса бросания элемента, надо назначить обработчик событий `drop` и `dragend`. В нем надо отменить поведение браузера по умолчанию, т.к. это может быть переход на другую страницу.

Объект `DataTransfer` хранит данные, отсылаемые в процессе перетаскивания. `DataTransfer` задается в событии `dragset` и используется в событии `drop`. Вызов `e.dataTransfer.setData(format, data)` устанавливает `mimetype` и данные нужные для перетаскивания.

Свойство `dataTransfer` имеет функцию `getData(format)` для получения сохраненных данных.

Свойство `dataTransfer` предоставляет возможность тонкой настройки отображения процесса переноса:

- `dataTransfer.effectAllowed` – эффект, поддерживаемый целевым эле-

ментом перетаскивания. Как правило, это значение задается обработчиком события `dragstart`. Может принимать следующие значения: `none`, `copy`, `copyLink`, `copyMove`, `link`, `linkMove`, `move`, `all` и `uninitialized`;

- `dataTransfer.dropEffect` – эффект, выбранный пользователем или целевым элементом. Может принимать следующие значения: `none`, `copy`, `link`, `move`;

- `dataTransfer.setDragImage(i, x, y)`. Вместо того чтобы использовать при перетаскивании полупрозрачную картинку по умолчанию

Алгоритм создания функций:

```
function dragStart(ev)
```

```
{
  ev.dataTransfer.effectAllowed='link';
  ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
  ev.dataTransfer.setDragImage(ev.target,0,0);
  return true;
}
```

```
function dragEnter(ev)
```

```
{
  var idelt = ev.dataTransfer.getData("Text");
  return true;
}
```

```
function dragOver(ev)
```

```
{
  var idelt = ev.dataTransfer.getData("Text");
  var id = ev.target.getAttribute('id');
  return false;
}
```

```
function dragEnd(ev)
```

```
{
  ev.dataTransfer.clearData("Text");
  return true
}
```

```

    }
function dragDrop(ev)
{
    var idelt = ev.dataTransfer.getData("Text");
    ev.target.appendChild(document.getElementById(idelt));
    ev.stopPropagation();
    return false;
}

```

Ниже приведен программный код для вызова соответствующих функций при возникновении событий:

```

<div id="source" class="container" ondragenter="return dragEnter(event)"
ondrop="return dragDrop(event)" ondragover="return dragOver(event)">
    <div id="firstDragElement" class="element" ondragstart="return drag-
Start(event)"
ondragend="return dragEnd(event)" draggable="true">Text 1</div>
    <div id="secondDragElement" class="element" ondragstart="return
dragStart(event)
" ondragend="return dragEnd(event)" draggable="true">Text 2</div>
    <div id="thirdDragElement" class="element" ondragstart="return drag-
Start(event)"
ondragend="return dragEnd(event)" draggable="true">Text 3</div>
</div>
<div id="target" class="container" ondragenter="return dragEnter(event)"
ondrop="return dragDrop(event)" ondragover="return dragO-
ver(event)"></div>

```

#### *Вставка элементов мультимедиа*

Для того, чтобы не загружать главную Web-страницу избыточным кодом, для каждого альбома создадим отдельную *HTML*-страницу, которую затем будем подгружать.

Таким образом, для выполнения задания, необходимо сделать следующее:  
- создать страницы альбомов и главную страницу;

- стилизовать страницы;
- создать *JavaScript* функции для: добавления трека в очередь воспроизведения, предварительного прослушивания трека, перехода между альбомами и воспроизведения трека из очереди.

Для того, чтобы пользователю не пришлось каждый раз заново создавать свой список воспроизведения, будем использовать `localStorage`.

Рассмотрим по шагам выполнение задания.

Шаг 1.

```
<body>
  <div id="tracks">
    <form id="albumtracks">
      <fieldset>
        <legend id="albumname">The Genius Hits The Road</legend>
        
        <table>
          <tr>
            <td>01-Alabamy Bound</td>
            <td><a class="addtoPLlink" onclick="addtoPL('Ray Charles-Alabamy
Bound',
audio/Ray Charles/The Genius Hits The Road/01-Alabamy Bound.mp3');
">Add to playlist</a></td>
            <td><a class="listenlink" onclick="testListen
('audio/Ray Charles/The Genius Hits The Road/01-Alabamy Bound.mp3');
">Listen</a></td>
          </tr>
          <tr>
            <td>02-Georgia On My Mind</td>
            <td><a class="addtoPLlink" onclick="addtoPL('Ray Charles-Georgia On My
Mind',
```

```

‘audio/Ray Charles/The Genius Hits The Road/02-Georgia On My Mind.mp3’);
“>Add to playlist</a></td>
    <td><a class=“listenlink” onclick=“testListen
(‘audio/Ray Charles/The Genius Hits The Road/02-Georgia On My Mind.mp3’);
“>Listen</a></td>
</tr>
<tr>
<td>03-Basin Street Blues</td>
<td><a class=“addtoPLlink” onclick=“addtoPL(‘Ray Charles-Basin Street
Blues’,
‘audio/Ray Charles/The Genius Hits The Road/03-Basin Street Blues.mp3’);
“>Add to playlist</a></td>
    <td><a class=“listenlink” onclick=“testListen
(‘audio/Ray Charles/The Genius Hits The Road/03-Basin Street Blues.mp3’);
“>Listen</a></td>
</tr>
<tr>
<td>04-Mississippi Mud</td>
<td><a class=“addtoPLlink” onclick=“addtoPL(‘Ray Charles-Mississippi
Mud’,
‘audio/Ray Charles/The Genius Hits The Road/04-Mississippi Mud.mp3’);
“>Add to playlist</a></td>
    <td><a class=“listenlink” onclick=“testListen
(‘audio/Ray Charles/The Genius Hits The Road/04-Mississippi Mud.mp3’);
“>Listen</a></td>
</tr>
.....
</body>

```

Обратите внимание, что сразу же заданы все атрибуты для дальнейшей стилизации, а также вызова *JavaScript*, а именно:

- addtoPL() – функция для добавления указанного трека в пользователь-

скую очередь воспроизведения;

- testListen() – функция для предварительного прослушивания трека (в течении 10 секунд, например).

Шаг 2. Создание *HTML*- документа для главной страницы:

```
<body onload="load();">
  <div id="header">
    <h1>Audio example</h1>
  </div>
  <div id="albums">
    <form id="albumform">
      <fieldset>
        <legend class="singer">Ray Charles</legend>
        <ul class="albumname">
          <li><a class="albumlink" onclick="loadAlbum('RC_hitstheroad.html');
"> The Genius Hits The Road</a></li>
          <li><a class="albumlink" onclick="loadAlbum('RC_smile.html')>Have
A Smile With Me</a></li>
        </ul>
      </fieldset>
    </form>
  </div>
  <div id="tracks">
    <iframe id="target" src=""></iframe>
  </div>
```

Поясним имена *JavaScript* – функций:

- load() – загрузка списка воспроизведения;
- loadAlbum() – загрузка внешнего *HTML* – документа (альбома).

Шаг 3. Создание минимально необходимых стилей:

```
#albums
```

```
{
```

```
float:left;
}
#tracks
{
float:left;
}
#audioPanel
{
clear:left;
}
#albumform
{
width:350px;
}
audio
{
width:100%
}
a.albumlink:hover
{
cursor:pointer;
font-style:italic;
font-weight:bolder;
}
#playlistpanel
{
border:3px double black;
}
#target
{
width:800px;
```

```
height:600px;
border:0px;
}
a.listenlink
{
margin:10px;
}
a.addtoPLink:hover, a.listenlink:hover
{
cursor:pointer;
font-style:italic;
font-weight:bolder;
}
#albumtracks
{
width:500px;
}
li.PL:hover
{
background-color:Blue;
cursor:pointer;
color:White;
}
body
{
background-color:#F7F7F7;
}
h1
{
text-align:center;
```

#### Шаг 4. Создание *JavaScript* функции

*Функция добавления трека в очередь воспроизведения:*

```
function addtoPL(name, path)
{
    localStorage[name] = path; /*добавляем в localStorage путь к аудио
    файлу*/
    window.parent.location.href=window.parent.location.href);
    /*поскольку iframe подгружает внешнюю html страницу, то после до-
    бавления трека в список, необходимо явно перезагрузить основную страни-
    цу*/
```

*Функция загрузки внешнего содержимого:*

```
function loadAlbum(path)
{
    var iframe = document.getElementById('target');
    iframe.setAttribute('src', path);
    sessionStorage["album"] = path; /*чтобы при перезагрузке родителя,
    текущая веб – страница альбома не исчезала, используем
    sessionStorage, для сохранения состояния*/
```

*Функция предварительного прослушивания:*

```
function testListen(path)
{
    var audio = document.createElement('audio'); /*создаем новый audio el-
    ement*/
    var div = document.getElementById('tracks');
    /*получаем доступ к контейнеру с треками*/
    audio.src= path; /*задаем источник для воспроизведения*/
    audio.controls = false;
    /*отключаем элементы управления аудио, фактически элемент не бу-
    дет отображаться*/
    div.appendChild(audio); /*добавляем созданный audio элемент контей-
    неру*/
```

```

audio.addEventListener('timeupdate', function()
    /*обработчик событий, вызываемый в течении всего воспроизведения
аудио, каждые 250 мс*/
    {
        if (audio.currentTime > 10) {audio.pause();}
        /*прерываем воспроизведение после 10 секунд прослушивания*/
    }, false);
audio.play(); /*начало воспроизведения аудио*/
Функция воспроизведения трека из пользовательского списка:
function playtrack(track)
    {

```

```
var b = false;
```

```
for(var I in localStorage)
```

```
{
```

```
/*в данном цикле мы находим текущий трек и «запоминаем» следующий*/
```

```
if (b) {localStorage['next'] = localStorage[i]; break; } if (I ==
```

```
track) { b=true; }
```

```
}
```

```
var audio = document.getElementById('audio'); /*получаем доступ к
элементу audio*/
```

```
audio.src = localStorage[track]; /*задаем источник воспроизведения*/
```

```
audio.controls = true;
```

```
audio.play(); /*начинаем воспроизведение*/
```

```
Функция загрузки списка воспроизведения:
```

```
function load()
```

```
{
```

```
if (sessionStorage["album"] != undefined)
```

```
/*если документ открывается впервые, либо после долгого перерыва,
то sessionStorage
```

```
не будет содержать информацию о последнем открытом альбоме*/
```

```
{
```

```

loadAlbum(sessionStorage[«album»]);
/*открываем последний альбом, к которому обратился пользователь,
таким образом
после принудительного обновления главной страницы в функции
adtoPL
пользователю не придется делать это самостоятельно*/
}
var audio = document.getElementById('audio');
/*получаем доступ к audio элементу*/ audio.addEventListener('ended',
function()
/*добавляем обработчик события – окончания текущего воспроизведения*/
{
playtrack('next');
/*инициуем воспроизведение следующего трека из пользовательского
списка*/
}, false);
/*переходим к загрузке пользовательского списка воспроизведения*/
var list = document.getElementById('playlist');
/*получаем доступ к списку воспроизведения*/
for(var i in localStorage) /*добавляем треки, сохраненные
пользователем*/
{
var element = document.createElement('li');
element.setAttribute('class', 'PL');
element.setAttribute('onclick', "playtrack( '"+i+"' )"); element.innerText = I;
if ( I != 'next')
{
list.appendChild(element);
}
}
}

```

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками:

процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт; устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### Задания

**Задание 1.** Реализовать функцию перемещения элементов Web-страницы между группой контейнерами, в следующих вариантах:

- все элементы могут свободно перемещаться между тремя контейнерами;
- ряд элементов, относящихся к одному классу нельзя перенести в один из контейнеров;
- элементы не могут быть возвращены в контейнер, в котором находились первоначально.

**Задание 2.** Создать HTML – документ, содержащий список альбомов и позволяющий формировать пользовательскую очередь воспроизведения мультимедийных материалов.

### Указания по порядку выполнения работы

При выполнении заданий к лабораторной работе использовать материал, приведенный в ее теоретической части.

#### Содержание отчета

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;
- 3) формулировки индивидуальных заданий и порядка их выполнения.

#### Контрольные вопросы

1. Охарактеризуйте алгоритм создания элементов перемещения внутри Web-формы.
2. При помощи каких тегов можно описать стилизацию элементов перемещения?
3. Дайте характеристику алгоритмам создания и использования JavaScript-функций в Web-формах.
4. Охарактеризуйте объект DataTransfer. Приведите его свойства.
5. Каким образом можно осуществить вставку элементов мультимедиа в Web-форму?
6. Характеристика и назначение JavaScript-функций при вставке объектов мультимедиа в Web-форму.

**Тема: Сценарии и приложения, выполняющиеся на стороне клиента**

### ЛАБОРАТОРНАЯ РАБОТА 3

#### HTML 5. РАБОТА С ГРАФИКОЙ. CANVAS. АНИМАЦИЯ

**Цель работы:** ознакомление с элементом `<canvas>`, ей `<canvas>` и создание простейших JavaScript-функций для генерации изображений в рамках элемента `<canvas>`; формирование базовых навыков рисования простых фигур на холсте.

#### Формируемые компетенции или их части:

ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных

систем и технологий.

## Теоретическая часть

### *Размещение <canvas> на странице*

Холст размещается на странице при помощи соответствующего тега:

```
<canvas id="lesson6" width="150" height="150"></canvas>
```

Атрибуты `width` и `height` не являются обязательными. По умолчанию высота и ширина холста равны 150 и 300 пикселям соответственно. При явном задании размеров холста его картинка масштабируется соответствующим образом.

По аналогии с такими новыми тегами как `<audio>` и `<video>`, элементы, размещенные в рамках парного тега `<canvas>` будут отображаться на странице в том случае, если используемый пользователем браузер не поддерживает `<canvas>`.

### *Стилизация <canvas>*

Стилизация холста не отличается от таковой для любого другого элемента Web-страницы. Могут быть заданы величины отступа, выравнивания, рамки и т.п.

Поскольку еще не создана функция по отрисовке содержимого холста, то и в браузере мы ничего не увидим. Создадим стиль для элемента `<canvas>`, задающий параметры границы вокруг элемента:

```
canvas { border: 3px double black;
```

### *Рисование на холсте*

`<canvas>` создаёт поверхность для рисования, предоставляющую один или более контекст отрисовки, который используется для создания отображаемого контента и манипуляций с ним.

Элемент `<canvas>` изначально пустой, и для того, чтобы что-либо отобразить, скрипту необходимо получить контекст отрисовки и рисовать уже на нём. Элемент `canvas` имеет *DOM-метод*, называемый `getContext`, и предназначенный для получения контекста отрисовки вместе с его функциями рисования. `getContext()` принимает один параметр – тип контекста. Осуществляется все вышесказанное следующим образом, в рамках *JavaScript*-функции:

```
var canvas = document.getElementById("lesson6");
var ctx = canvas.getContext("2d")
```

Первая строка приведенного фрагмента создает переменную, фактически являющуюся созданным холстом. Вторая – получает доступ к контексту рисования.

Рассмотрим пример создания изображения. Пусть это будет – прямоугольник багрового цвета.

Для этого, создадим *JavaScript* – функцию:

```
function createImage()
{
    var canvas = document.getElementById("lesson6");
    var ctx = canvas.getContext("2d");
}
```

Прямоугольную закрашенную область можно нарисовать при помощи функции `fillRect (x,y,w,h)`, где `x` и `y` – координаты левой верхней вершины прямоугольника (по горизонтали и вертикали соответственно), а `w` и `h` – значения ширины и высоты прямоугольника, соответственно.

Функция `strokeRect (x,y,w,h)` рисует границы прямоугольника, `clearRect (x,y,w,h)` – очищает заданную прямоугольную область.

Функция `fillStyle` задает цвет рисования.

Таким образом, функция принимает вид:

```
function createImage()
{
    var canvas = document.getElementById("lesson6");
    var ctx = canvas.getContext("2d");
    ctx.fillStyle = "rgb(100,0,0)"
    ctx.fillRect (20, 10, 50, 70);
}
```

Функция готова, холст уже давно размещен на странице, осталось обеспечить выполнение функции. Для того, чтобы изображение обрисовывалось в момент загрузки страницы, добавим в тег `<body>` вызов функ-

ции `createImage` при загрузке. HTML-код будет выглядеть следующим образом:

```
<body onload="createImage();">
<canvas id="text" width="150" height="150"></canvas>
</body>
```

### *Рисование фигур*

Каждый треугольник может быть нарисован последовательным вызовом трех методов `lineTo`.

Шаг 1. Создание пути для отрисовки первого треугольника:

```
function draw(){
    var canvas = document.getElementById('text');
    var ctx = canvas.getContext('2d');
    ctx.beginPath();
    ctx.moveTo(20,20); // задаем начальную точку для "пера"
    ctx.lineTo(20, 120); // сторона А
    ctx.lineTo(120,120); // сторона В
    ctx.lineTo(20,20); // сторона С
    ctx.stroke(); // рисуем незаполненную фигуру
```

Шаг 2. Создание залитого треугольника отличается от предыдущего шага незначительно, а именно, вызовом метода `fill`, вместо `stroke`. Добавим соответствующий код в функцию `draw`:

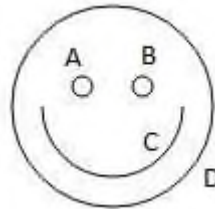
```
function draw(){
    var canvas = document.getElementById('lesson6');
    var ctx = canvas.getContext('2d');
    ctx.beginPath();
    ctx.moveTo(20,20); // задаем начальную точку для "пера"
    ctx.lineTo(20, 120); // сторона А
    ctx.lineTo(120,120); // сторона В
    ctx.lineTo(20,20); // сторона С
    ctx.stroke(); // рисуем незаполненную фигуру
```

```
ctx.beginPath();
ctx.moveTo(40,20); // задаем начальную точку для отрисовки второго
треугольника
```

```
ctx.lineTo(140, 20); // сторона D
ctx.lineTo(140,120); // сторона E
ctx.lineTo(40,20); // сторона F
ctx.fill(); // рисуем залитую цветом фигуру
```

### *Рисование дуг*

Например, используем процесс построения дуг на следующем изображении. В нем можно выделить всего четыре дуги, которые необходимо нарисовать:



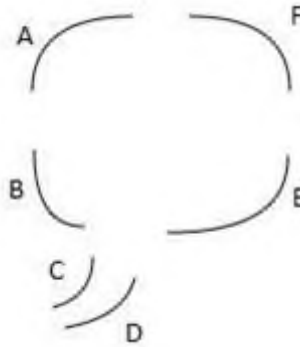
Основная сложность этого задания в переносе "пера" в начальные позиции для рисования, поэтому обратите внимание на координаты, указанные в методах `moveTo` и `arc`:

```
function draw(){
var canvas = document.getElementById('lesson6');
var ctx = canvas.getContext('2d');
ctx.beginPath();
ctx.moveTo(65,65); // начальная позиция для рисование
ctx.arc(60,65,5,0,Math.PI*2,true); // дуга A
ctx.moveTo(95,65); // перенос "пера" в позицию для отрисовки дуги B
ctx.arc(90,65,5,0,Math.PI*2,true); // дуга B
ctx.moveTo(110,75); // перенос "пера" для отрисовки дуги C
ctx.arc(75,75,35,0,Math.PI,false); // дуга C
ctx.moveTo(125,75); // перенос "пера" для отрисовки дуги D
ctx.arc(75,75,50,0,Math.PI*2,true); // дуга D
```

`ctx.stroke();` // отрисовка незаполненной фигуры

### *Кривые Безье*

Отметим только, что для отрисовки каждой "половинки сердца" используется три кривые (метод `bezierCurveTo`).



Если рисовать кривые последовательно от А к F, то понадобится только один вызов функции `moveTo` для задания начальных координат для первой кривой (А). Как и в предыдущем примере нужно уделить внимание задаваемым координатам для кривых:

```
function draw(){
    var canvas = document.getElementById('lesson6');
    var ctx = canvas.getContext('2d');
    ctx.beginPath();
    ctx.moveTo(75,25); // задание начальных координат
    ctx.quadraticCurveTo(25,25,25,62.5); // кривая А
    ctx.quadraticCurveTo(25,100,50,100); // кривая В
    ctx.quadraticCurveTo(50,120,30,125); // кривая С
    ctx.quadraticCurveTo(60,120,65,100); // кривая D
    ctx.quadraticCurveTo(125,100,125,62.5); // кривая Е
    ctx.quadraticCurveTo(125,25,75,25); // кривая F
    ctx.stroke(); // отрисовка незаполненной фигуры
```

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт;

устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### Задания

**Задание 1.** Разместите элемент `<canvas>` на странице.

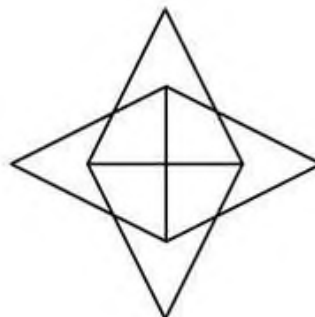
**Задание 2.** Создать файл стилей для `<canvas>`.

**Задание 3.** При помощи JavaScript-функции реализовать генерацию следующих изображений на странице:

1. Рисование путей

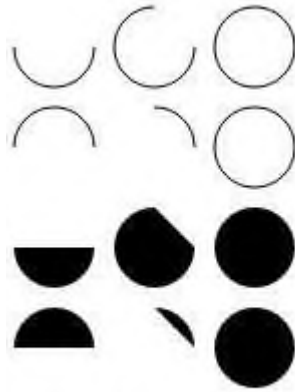


а)



б)

## 2. Дуги



а)

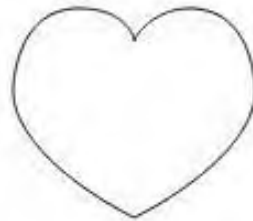


б)

## 3. Кривые Безье



а)



б)

## 4. Комплексное задание



### **Указания по порядку выполнения работы**

При выполнении заданий к лабораторной работе следует использовать материал и примеры, приведенные в ее теоретической части.

### **Содержание отчета**

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;
- 3) формулировки индивидуальных заданий и порядка их выполнения.

### **Контрольные вопросы**

1. При помощи какого тега осуществляется размещение холста на странице?
2. С какой целью в HTML-документах проводится стилизация холста?
3. Охарактеризуйте процесс рисования на холсте. Назовите функции, которые для этого используются.
4. Приведите алгоритм рисования простейших фигур на холсте.
5. Каким образом осуществляется рисование дуг?
6. Алгоритм рисования «кривой Безье».

## **Тема: Введение в C# и платформу Visual Studio.Net**

### **ЛАБОРАТОРНАЯ РАБОТА 4**

#### **ЯЗЫК C#. СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКЕ C#**

**Цель работы:** знакомство со средой разработки приложений Microsoft Visual Studio.NET и структурой программы на языке C# (для консольного приложения).

#### **Формируемые компетенции или их части:**

ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий.

## Теоретическая часть

Microsoft Visual Studio .NET – это интегрированная среда разработки для создания, документирования, запуска и отладки программ, написанных на языках .NET.

Эта среда разработки является открытой языковой средой, в которую наряду с языками программирования, изначально включенными в среду – C++, C#, J#, Visual Basic, – в нее могут добавляться любые языки программирования, компиляторы которых создаются сторонними разработчиками. Необходимым условием для включения языков в среду Visual Studio .NET является использование единого каркаса – платформы Framework.Net.

Платформа Framework.NET позволяет:

- легко использовать компоненты, разработанных на различных языках;
- разрабатывать единое приложение из нескольких частей на разных языках.

Для разработки приложений в Visual Studio.NET используются проекты.

Проект (Project) – это основная единица. Сначала он должен выбрать тип проекта, после чего Visual Studio создает каркас проекта в соответствии с выбранным типом.

Проект состоит из классов, собранных в одном или нескольких пространствах имен. Пространства имен (Namespaces) позволяют структурировать проекты, содержащие большое число классов, объединяя в одну группу близкие классы.

Несколько проектов могут объединяться в решение (Solution), которое также может включать ресурсы, необходимые этим проектам.

Конечным результатом работы, получаемым после компиляции исходного программного кода, является решение, а с точки зрения CLR (Common Language Runtime – общезыковой среды исполнения) – сборка (assembly), содержащая PE файл, т.е. модуль в формате исполняемого файла PE (Portable Executable) для 32-разрядной ОС Windows либо DLL (Dynamic Link Library)

файл.

Visual Studio.Net предлагает большое разнообразие возможных типов проектов. Для ознакомления с основами языка C# в основном будут использованы консольные приложения без привлечения привычных в Windows возможностей пользовательского интерфейса.

В C# нет глобальных функций, поэтому по умолчанию при создании консольного приложения объявляется класс Program, который содержит функцию static Main(), служащую начальной точкой выполнения программы. Функция Main может быть объявлена без параметров или с параметром, представляющий собой массив строк.

Также программа на C# может содержать комментарии и атрибуты.

В C# используются однострочные и многострочные комментарии. Однострочный комментарий начинается с пары символов " // ", и весь текст до конца строки является комментарием. Многострочный комментарий начинается с пары символов /\* и заканчивается \*/.

Атрибут – средство добавления декларативной информации к элементам программного кода. Назначение атрибутов: организация взаимодействия между программными модулями, дополнительная информация об условиях выполнения кода, управление сериализацией (т.е. правила сохранения информации), отладка, и др. Атрибуты могут быть как стандартными, так и заданными пользователем. Стандартные атрибуты используются CLR и влияют на то, как будет выполняться проект. Атрибут размещается в скобках " [] ".

Стандарт языка C++ включает следующий набор фундаментальных типов:

- логический тип (bool);
- символьный тип (char);
- целые типы. Они могут отличаться размером: short, int, long, а также могут быть знаковыми (signed) или беззнаковыми (unsigned).
- типы с плавающей точкой. Они также могут отличаться размерами: float, double и long double;
- тип void указывает на отсутствие информации.

К конструируемым типам относятся следующие: указатели (например, `char*`); ссылки (например, `char&`); массивы (например, `char[]`).

Также язык позволяет разработчику конструировать собственные типы: перечислимые типы (`enum`); структуры (`struct`).

В языке *C#* все типы можно рассматривать и под другим ракурсом, разделив их на четыре категории: типы-значения (`value`); ссылочные (`reference`); указатели (`pointer`); тип `void`.

Для ссылочного типа значение задает ссылку на область памяти в "куче" (`heap`), где расположен соответствующий объект. Для типа-значения значением являются собственно данные, а память для них выделяется в стеке.

Логический, арифметический, структуры, перечисление относятся типам-значениям. Массивы, строки и классы относятся к ссылочным типам.

И ссылочные, и обычные типы являются производными от базового класса `object`. В тех случаях, когда обычный тип должен вести себя как объект, создается оболочка (`wrapper`), которую можно рассматривать как ссылочный объект, помещенный в кучу, и в нее копируется значение переменной обычного типа. Оболочка автоматически помечается таким образом, что система знает, какое значение она содержит. Этот процесс называется упаковкой (`boxing`), а обратный процесс – распаковкой (`unboxing`).

Упаковка происходит автоматически, для этого нужно только присвоить значение обычного типа переменной типа `object`. Упаковка и распаковка позволяют обрабатывать любой тип как объект. Например, в выражении `7.ToString()`; целое число `7` упаковывается путем вызова функции `Int32.ToString()`.

Массивы в *C#* могут быть многомерными (`multidimensional`) или невыровненными (`jagged`). Более сложные структуры данных такие, как стек и хеш-таблица определены в пространстве имен `System.Collections`.

### ***Выражения и операторы C#***

Выражения строятся из операндов – констант, переменных, функций, – объединенных знаками операций и скобками. При вычислении выражения

определяется его значение и тип.

Таблица 4.1 – Список операций C#

Категория операций	Операции
Арифметические	+ - * / %
Логические (boolean и побитовые)	&   ^ ! ~ &&
Строковые	+
Инкремент и декремент	++ --
Сдвиг	>> <<
Сравнение	== != < > <= >=
Присвоение	= += -= *= /= %= &=  = ^= <<= >>=
Обращение к члену класса	.
Индексация	[]
Приведение типа (Cast)	()
Условие	?:
Создание объекта	new()
Информация о типе	is sizeof typeof
Управление исключениями	checked unchecked
Косвенности и адресации	* -> [] &

Имя и тип переменной задаются при ее объявлении и остаются неизменными в течение всего времени ее жизни. Особенностью языка C# является требование обязательной инициализации переменной до начала ее использования. Попытка использовать неинициализированную переменную приводит к ошибкам, обнаруживаемым еще на этапе компиляции.

По используемым выражениям и операторам C# похож на C++. Так в программах на C# используются такие операторы как: оператор присваивания (=); составной оператор ({}); операторы выбора: if-else и switch; операторы цикла: for, while, оператор; операторы break и continue; оператор return; оператор перехода goto

Кроме того, введены несколько новых инструкций. Например, оператор foreach позволяет получить доступ ко всем элементам массива или коллекции поочередно, в порядке возрастания индексов. Его синтаксис:

*foreach (тип идентификатор in контейнер) оператор*

В C# процедуры и функции существуют только как методы некоторого класса, они не определены вне класса. Роль библиотек процедур и функций

выполняют библиотеки классов. Библиотека классов Framework Class Library (FCL), доступная в языке C#, существенно расширяет возможности языка.

### *Классы и методы в C#*

В Visual Studio.Net, и в C# в частности, любая программная система рассматривается как совокупность классов, объединенных в проекты, пространства имен, решения.

Описание класса имеет следующий синтаксис:

```
[атрибуты][модификаторы]class имя_класса[:список_родителей]
    {тело_класса}
```

В теле класса могут быть объявлены: константы; поля; конструкторы и деструкторы; методы; события; делегаты; классы (структуры, интерфейсы, перечисления).

Поля класса синтаксически являются обычными переменными (объектами) языка. Их описание удовлетворяет обычным правилам объявления переменных. Поля характеризуют свойства объектов класса.

Методы класса синтаксически являются обычными процедурами и функциями языка. Методы содержат описания операций, доступных над объектами класса. Методы, называемые свойствами являются специальной синтаксической конструкцией, предназначенной для обеспечения эффективной работы с классами.

Конструктор представляет собой специальный метод класса, позволяющий создавать объекты класса. Его имя должно совпадать с именем класса. Если разработчик не определяет конструктор класса, то к классу автоматически добавляется конструктор по умолчанию - конструктор без аргументов.

Делегат в C# представляет собой описание специального случая класса и задает определение функционального типа (класса) данных. Экземплярами класса являются функции. Каждый делегат описывает множество функций с заданной сигнатурой. Каждая функция (метод), сигнатура которого совпадает с сигнатурой делегата, может рассматриваться как экземпляр класса, заданного делегатом. Синтаксис объявления делегата имеет следующий вид:

*[<спецификатор доступа>] delegate <тип результата>  
<имя класса> (<список аргументов>);*

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт; устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц, среда программирования Visual Studio .Net.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### Задание

Разработать консольное приложения для вычисления корней квадратного уравнения. Требуется разработать приложение, которое по заданным значениям коэффициентов  $a$ ,  $b$  и  $c$  квадратного уравнения (значения вводятся с клавиатуры пользователем) вычисляет и отображает на экране корни уравнения.

Для данного приложения потребуются следующие методы:

- string Console.ReadLine() – чтение строки символов из входного потока;
- double Convert.ToDouble(string) – преобразование строки символов в число с плавающей запятой двойной точности;

- `double Math.Sqrt(double)` – извлечение квадратного корня числа.

### Указания по порядку выполнения работы

#### *Создание проекта консольного приложения*

1. Первый шаг при разработке любого приложения – создание проекта. Для этого через разделы меню: File → New → Project запускается панель создания нового проекта.

2. Из списка предлагаемых шаблонов необходимо выбрать проект консольного приложения (Console Application).

3. После создания проекта должно появиться следующее окно, содержащее текст программы Program.cs на языке C#.

В данной программе объявление `using System` дает возможность ссылаться на классы, которые находятся в пространстве имен System, так что их можно использовать, не добавляя "System." перед именем типа.

Пространство имен System содержит много полезных классов. Одним из них является и класс Console, который используется при создании консольных приложений.

Класс проекта – Program размещен в пространстве имен, имеющем по умолчанию то же самое имя (HelloWorldDemo), что и решение, содержащее в данном примере единственный проект.

Поскольку в C# нет глобальных функций, поэтому в данном примере объявляется класс Program, который содержит функцию `static Main()`, служащую начальной точкой выполнения программы. Функция Main может быть объявлена без параметров или с параметром, представляющий собой массив строк. Так как функция Main является точкой входа, она должна быть статической (`static`) функцией, т.е. она не связана с конкретным объектом класса, в котором объявлена.

4. Добавим внутри функции Main следующую строку:

***Console.WriteLine("Hello world!");***

которая использует метод WriteLine класса Console для вывода строки "Hello world!".

### *Компиляция проекта*

Для этого можно использовать либо меню (Build → Build Имя проекта) или сочетание клавиш <Shift+F6>. При наличии сообщений об ошибках необходимо их исправить и скомпилировать проект снова.

### *Запуск приложения*

Для можно использовать либо меню (Debug → Start Without Debugging) или сочетание клавиш <Ctrl+F5>.

### *Усложним приложение*

Для этого добавим в функцию Main следующий код для вывода в консольное окно содержимого массива целых чисел:

```
static void Main(string[] args)
{
    int[] ArInt = { 0, 9, 1, 8, 2, 7, 3, 6, 4, 5 };
    Console.WriteLine("Array contains elements:");
    foreach (int item in ArInt)
    {
        Console.Write("{0} ", item);
    }
    Console.WriteLine("");
}
```

### **Содержание отчета**

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;
- 3) формулировки индивидуальных заданий и порядка их выполнения.

### **Контрольные вопросы**

1. Для чего предназначена платформа Framework.Net?
2. Что представляет собой проект в Visual Studio.Net?
3. Сформулируйте определение понятия «атрибут».
4. Перечислите и охарактеризуйте типы данных, которые используются

в языке C++.

5. Типы каких четырех категорий выделяют в C#?
6. Назовите основные операции, которые применяются в C#.
7. Какие основные операторы используются в C#?
8. Как можно описать класс в C#?

## **Тема: Введение в C# и платформу Visual Studio.Net**

### **ЛАБОРАТОРНАЯ РАБОТА 5 РАБОТА С МАССИВАМИ И СТРОКАМИ**

**Цель работы:** продолжить знакомство со средой разработки приложений Microsoft Visual Studio.NET.; изучить особенности работы с массивами и строками в C#, а также с реализацией интерфейсами в C#.

#### **Формируемые компетенции или их части:**

1. ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий.

#### **Теоретическая часть**

##### *Массивы*

Язык C# предоставляет средства для работы с одномерными массивами, массивами массивов и многомерными массивами.

Ниже представлены способы объявления ссылок на массивы различной размерности и конфигурации.

// Объявление массива размерности 3 из элементов типа int.

// Фактически это массив составляющих, представляющих собой массивы

// размерности 2 одномерных массивов элементов типа int.

// Размеры всех составляющих массивов одного уровня равны

// между собой ("прямоугольный" массив).

int[, ] ar1;

// Объявление одномерного массива из

// одномерных массивов элементов, каждый из которых является

```

// одномерным массивом элементов типа int.
int[][] ar2;
// Объявление одномерного массива составляющих, каждый из которых является
// двумерным массивом массивов элементов типа int.
// При этом у всех составляющих могут быть разные размеры
int[,] ar3;
// Объявление двумерного массива составляющих, каждая из которых является
// одномерным массивом элементов типа int.
// При этом у всех составляющих могут быть разные размеры.
int[,] ar4;

```

Рассмотренные примеры демонстрируют определение двух различных категории массивов: прямоугольные массивы, ломаные (jagged) массивы.

Объявление массива может быть совмещено с инициализацией. При объявлении с отложенной инициализацией сам массив не формируется, создается только ссылка на массив, имеющая неопределенное значение Null. Поэтому пока массив не будет реально создан и его элементы инициализированы, использовать его нельзя.

Возможны два варианта объявления массива, совмещенного с инициализацией.

В первом случае инициализация является явной и задается списком констант, например: `int[] odd = {1, 3, 5, 7, 9}`.

Во втором случае создание и инициализация массива реализуется вместе с вызовом конструктора массива. Например: `int[] even = new int[5]`.

Все массивы являются потомками класса `System.Array`, который в свою очередь наследует ряд сов: `ICloneable`, `IList`, `ICollection`, `IEnumerable`, а, следовательно, реализует все их методы и свойства. Кроме того, класс `Array` имеет большое число собственных методов и свойств.

Благодаря наследованию от класса `System.Array`, для массивов определены такие встроенные методы копирования, поиска, обращения, сортировки. Массивы можно рассматривать также как коллекции и использовать цикл `foreach` для перебора его элементов.

Таблица 5.1 – Основные свойства класса `System.Array`

Свойство	Описание
<code>IsFixedSize</code>	Возвращает <code>true</code> , если массив является статическим
<code>IsReadOnly</code>	Для всех массивов возвращает значение <code>false</code>
<code>IsSynchronized</code>	Возвращает <code>true</code> или <code>false</code> , в зависимости от того, установлена ли синхронизация доступа для массива
<code>Length</code>	Число элементов в массиве
<code>Rank</code>	Размерность массива

Таблица 5.2 – Основные методы класса `System.Array`

Метод	Описание
<code>BinarySearch</code>	Бинарный поиск элементов
<code>Clear</code>	Выполняет инициализацию элементов. Числовые элементы становятся нулевыми, а строковые принимают значение <code>Null</code>
<code>Clone</code>	Клонирование массива
<code>Copy</code>	Копирование части или всего массива в другой массив
<code>CopyTo</code>	Копируются все элементы одномерного массива в другой одномерный массив, начиная с заданного индекса
<code>CreateInstance</code>	Создание экземпляра массива
<code>GetEnumerator</code>	Возвращает интерфейс <code>IEnumerator</code> . Необходим для использования в цикле <code>foreach</code>
<code>GetLength</code>	Возвращает число элементов массива по указанному измерению
<code>GetLowerBound, GetUpperBound</code>	Возвращает нижнюю и верхнюю границу по указанному измерению. Для массивов нижняя граница всегда равна нулю.
<code>GetValue SetValue</code>	Возвращает или устанавливает значение элемента массива с указанными индексами.
<code>IndexOf</code>	Индекс первого вхождения образца в массив
<code>LastIndexOf</code>	Индекс последнего вхождения образца в массив
<code>Reverse</code>	Сортировка одномерного массива в обратном порядке
<code>Sort</code>	Сортировка массива

### Строки

В языке `C#` определен класс `char[]`, и его можно использовать для пред-

ставления строк постоянной длины. Однако массив `char[]` – это обычный массив, поэтому его нельзя инициализировать строкой символов. В C# не определено преобразование из класса `char[]` в класс `String`. У `String` есть динамический метод `ToCharArray`, задающий подобное преобразование в `char[]`.

Объекты класса `String` объявляются с явной или отложенной инициализацией, с явным или неявным вызовом конструктора класса. Чаще всего, при объявлении строковой переменной конструктор явно не вызывается, а инициализация задается строковой константой. У класса `String` достаточно много конструкторов, которые позволяют сконструировать строку из: символа, повторяющегося указанное число раз; массива символов `char[]`; части массива символов.

Методы класса `String` позволяют выполнять вставку, удаление, замену, поиск вхождения подстроки в строку. Класс `String` наследует методы класса `Object`, а также методы четырех интерфейсов: `Comparable`, `Cloneable`, `Convertible`, `Enumerable`.

Таблица 5.3 – Члены класса

Метод	Описание
<code>Length</code>	Это свойство возвращает длину указанной строки
<code>Insert</code>	Вставляет подстроку в заданную позицию
<code>Remove</code>	Удаляет подстроку в заданной позиции
<code>Replace</code>	Заменяет подстроку в заданной позиции на новую подстроку
<code>Substring</code>	Выделяет подстроку в заданной позиции
<code>IndexOf</code> , <code>IndexOfAny</code> , <code>LastIndexOf</code> , <code>LastIndexOfAny</code>	Определяются индексы первого и последнего вхождения заданной подстроки или любого символа из заданного набора
<code>StartsWith</code> , <code>EndsWith</code>	Возвращается <code>true</code> или <code>false</code> , в зависимости от того, начинается или заканчивается строка заданной подстрокой
<code>PadLeft</code> , <code>PadRight</code>	Выполняет заполнение нужным числом пробелов в начале и в конце строки
<code>Trim</code> , <code>TrimStart</code> , <code>TrimEnd</code>	Удаляются пробелы в начале и в конце строки, либо только с одного конца
<code>ToCharArray</code>	Преобразование строки в массив символов

Класс `String` не разрешает изменять существующие объекты. Для этой цели имеется другой класс – класс `StringBuilder`, который позволяет исправить данный недостаток. Этот класс принадлежит к изменяемым классам и его можно найти в пространстве имен `System.Text`.

Объекты этого класса объявляются с явным вызовом конструктора класса. Конструктор класса перегружен, и наряду с конструктором без параметров, создающим пустую строку, имеется набор конструкторов, которым можно передать две группы параметров:

- первая группа позволяет задать строку или подстроку, значением которой будет инициализироваться создаваемый объект класса `StringBuilder`;
- вторая группа параметров позволяет задать размер объекта, т. е. объем памяти, отводимой для экземпляру класса `StringBuilder`.

Примеры конструкторов:

- `public StringBuilder(string str, int size)`. Параметр `str` задает строку для инициализации, `size` – размер объекта;
- `public StringBuilder(int cursize, int maxsize)`. Параметры `cursize` и `maxsize` задают начальный и максимальный размер объекта;
- `public StringBuilder(string str, int start, int len, int size)`. Параметры `str`, `start`, `len` задают строку инициализации, `size` – размер объекта.

Над строками этого класса определены (также как и для строк класса `String`) операции: присваивание (`=`); две операции проверки эквивалентности (`==`) и (`!=`); взятие индекса (`[]`).

Операция конкатенации (`+`) не определена над строками класса `StringBuilder`, ее роль играет метод `Append`, добавляющий новую строку в конец существующей.

### *Интерфейсы*

Интерфейс представляет собой полностью абстрактный класс, все методы которого абстрактны. Однако методы интерфейса объявляются без указания модификатора доступа, и класс, наследующий интерфейс, обязан пол-

ностью реализовать все методы интерфейса. В этом – отличие от класса, наследующего абстрактный класс, где потомок может реализовать лишь некоторые методы родительского абстрактного класса, оставаясь абстрактным классом.

Интерфейс позволяет описывать некоторые желательные свойства, которыми могут обладать объекты разных классов.

Среди интерфейсов, встроенных в библиотеку базовых классов *.NET*, можно особо выделить такие как:

- *IEnumerable* – для работы с наборами объектов, в т.ч. с использованием оператора *foreach*;

- *IClonable* – копирование объектов;

- *IComparable* – сравнение и сортировка объектов.

Пространство имен *System.Collections*, предназначенное для работы с наборами объектов, поддерживает интерфейсы:

- *Collection* – определяет общие характеристики класса набора элементов);

- *IComparer*, *IDictionary* – позволяет представлять содержимое объекта в виде пар «имя-значение»;

- *IDictionaryEnumerator* – нумерация содержимого объекта, поддерживающего *IDictionary*;

- *IEnumerable*,

- *IEnumerator*;

- *IHashCodeProvider* – возвращает хэш-код с помощью выбранного алгоритма хэширования);

- *IList* – обеспечивает методы добавления, удаления и индексирования элементов в списке объектов.

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт;

устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц, среда программирования Visual Studio .Net.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### **Задание**

Разработать консольное приложение для ввода с клавиатуры массива строк и поиска среди них строк, содержащих заданный строковый фрагмент.

Для поиска необходимой строки используйте метод `IndexOf (string findThisString)` для строковых элементов массива. Метод возвращает позицию начала искомой подстроки от начала строки, либо значение «-1» при отсутствии соответствия.

### **Указания по порядку выполнения работы**

#### *1. Работа с массивами.*

Следующие две программы демонстрируют, каким образом выполняется инициализация и работа с прямоугольными и ломаными массивами в C#.

*Прямоугольный массив:*

```
using System;
```

```

using System.Collections.Generic;
using System.Text;
namespace HelloWorldDemo
{
class Program
{
static void Main(string[] args)
{
// Прямоугольный многомерный массив
int[,] multMatr;
ultMatr = new int[10,10];
// Заполнение массива 9 на 9:
for (int i = 1; i <= 9; i++)
for (int j = 1; j < 9; j++)
multMatr[i, j] = i * j;
// Вывод элементов многомерного массива
for (int i = 1; i <= 9; i++)
{
for (int j = 1; j <= 9; j++)
{
Console.Write(multMatr [i, j] + "\t");
}
Console.WriteLine();
}
}
}
}
}

```

*Ломаный массив*

```

using System;
using System.Collections.Generic;
using System.Text;

```

```
namespace HelloWorldDemo
{
class Program
{
static void Main(string[] args)
{
// Ломаный многомерный массив из пяти внутренних массивов разного
размера
int[][] JagArr = new int[10][];
// Инициализация генератора случайных чисел
Random rand = new Random();
// Динамическое создание ломаного массив
for (int i = 0; i < JagArr.Length; i++)
{
JagArr[i] = new int[i + rand.Next(10)];
}
// Вывод строк на консоль
// Каждый элемент по умолчанию имеет значение, равное
for (int i = 0; i < 10; i++)
{
// Свойство Length массива возвращает его размер
Console.WriteLine("Length of row {0} is {1}:\t", i, JagArr[i].Length);
for (int j = 0; j < JagArr[i].Length; j++)
{
Console.WriteLine(JagArr[i][j] + " ");
}
Console.WriteLine();
}
}
}
```

## 2. Свойства и методы класса *Array*.

Следующий пример демонстрирует использование некоторых из приведенных свойств и методов класса *Array*:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace HelloWorldDemo
{
class Program
{
static void Main(string[] args)
{
// МАССИВ СИМВОЛЬНЫХ СТРОК
string[] Brands = new string[]
{"Audi",
"BMW",
"Buick",
"Chevrolet",
"Citroen",
"Dodge",
"Ferrari",
"Fiat",
"Ford",
"Honda",
"Hyundai",
"Cherokee",
"Cherry",
"Lada",
"Lamborghini",
"Lincoln",
"Mazda",
```

```

    "Mercedes",
    "Mitsubishi",
    "Nissan",
    "Opel",
    "Peugeot",
    "Plymoth",
    "Pontiac",
    "Renault",
    "Rover",
    "Saab",
    "Subaru",
    "Suzuki",
    "Toyota",
    "Volkswagen",
    "Volvo" };

    // Вывод марок автомобилей в соответствии с порядком элементов в
    массиве
Console.WriteLine("Here is the array of car brands:"); for (int i
= 0; i < Brands.Length; i++)
    Console.Write(Brands[i] + "\t");
    Console.WriteLine("\n\n");

    // Сортировка элементов в обратном порядке
    Array.Reverse(Brands);

    // Вывод автомарок
Console.WriteLine("Here is the list once reversed:");
for (int i = 0; i < Brands.Length; i++)
    Console.Write(Brands[i] + "\t");
Console.WriteLine("\n\n");

    // Остаются только первый и последний
    Console.WriteLine("Only two remain: ");
array.Clear(Brands, 2, Brands.Length-2); for

```

```

    (int i = 0; i < Brands.Length; i++)
    {
        Console.Write(Brands[i] + "\t\n");
    }
}
}
}
}

```

### 3. Использование класса *StringBuilder*:

```

using System;
using System.Collections.Generic;
using System.Text;
namespace HelloWorldDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            StringBuilder WordList = new StringBuilder("Каждый Охотник ");
            WordList.Append("Желает Знать Где ");
            Console.WriteLine(WordList);
            WordList.Append("Сидит Фазан");
            Console.WriteLine(WordList);
            // Сделать все буквы прописными
            string Spectrum = WordList.ToString().ToUpper();
            Console.WriteLine(Spectrum);
        }
    }
}

```

### 4. Использование класса *ArrayList*

Использование класса `ArrayList` из пространства имен `System.Collections` позволяет эффективно реализовать работу с массивами.

вами объектов, поскольку многие возможности, необходимые для этого реализованы изначально, в частности методы вставки, удаления и нумерации элементов.

Для использования возможностей `ArrayList` используется не обычное наследование, а модель включения в виде делегирования вызовов на выполнение различных действий классу производному от `ArrayList`:

```
public class NBooks : IEnumerable
{
    // nbList – внутренний класс, который будет делать всю работу
    private ArrayList nbList;
    // Создаем объект класса nbList при помощи конструктора NBooks
    public NBooks() {nbList = new ArrayList();}
    // Реализуем нужные нам методы для приема вызовов извне и передачи их
    nbList
        // Метод для вставки объекта NBook
        public void AddNBook(NBook nb)
            { nbList.Add(nb); }
        // Метод для удаления объекта NBook
        public void RemoveNBook(int nbToRemove)
            {nbList.RemoveAt(nbToRemove); }
        // Свойство, возвращающее количество объектов NBook
        public int NBookCount
            { get { return nbList.Count; } }
        // Метод для очистки объекта — удаления всех объектов NBook
        public void ClearAllNBooks()
            { nbList.Clear(); }
        // Метод, который отвечает на вопрос — есть ли уже в наборе такой объект
        NBook
        public bool NBookIsPresent(NBook c)
            { return nbList.Contains(c); }
```

```
// Все, что связано с реализацией IEnumerator, перенаправляется в nbList
    public IEnumerator GetEnumerator()
    { return nbList.GetEnumerator(); }
}
```

Реализация класса NBook и код программы,  
щей класс NBooks представлены ниже.

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Text;
namespace NBooks
{
    public class NBook
    {
// Конструктор класса NBook
        public string Model;
        public string CPU_model;
        public int CPU_clock;
        public int RAM_size;
        public NBook(string mname, string CPU, int Clock, int RAM)
        {
            Model = mname;
            CPU_model = CPU;
            CPU_clock = Clock;
            RAM_size = RAM;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
```

```
NBooks nbLot = new NBooks();
// Создание списка объектов NBook
    nbLot.AddNBook( new NBook("ASUS A7Sn","Intel Core 2 Duo T8300",
2400, 2048));
    nbLot.AddNBook( new NBook("Acer Aspire 5530G-803G25Mi","AMD
Turion X2 Ultra ZM80", 2100, 3072));
    nbLot.AddNBook( new NBook("Fujitsu Amilo Si 2636", "Intel Core 2 Duo
T8300", 2400, 2048));
    nbLot.AddNBook( new NBook("HP Pavilion tx2650er", "AMD Turion X2
Ultra ZM82", 2200, 4096));
// Выводим информацию о каждом объекте при помощи конструкции foreach
    Console.WriteLine("You have {0} in the lot: \n", nbLot.NBookCount);
    foreach (NBook nb in nbLot)
    {
        Console.WriteLine("Model: {0}", nb.Model);
        Console.WriteLine("CPU: {0}", nb.CPU_model);
        Console.WriteLine(" {0} GHz", nb.CPU_clock);
        Console.WriteLine("RAM: {0} GB\n", nb.RAM_size);
    }
// Удаляем один из ноутбуков
    nbLot.RemoveNBook(3);
    Console.WriteLine("You have {0} in the lot.\n", nbLot.NBookCount);
// Добавляем еще один ноутбук и проверяем его наличие в наборе
    NBook temp = new NBook("ASUS M51Ta", "AMD Turion™ X2 Ultra
ZM84", 2300, 4096);
    nbLot.AddNBook(temp);
    if(nbLot.NBookIsPresent(temp))

        Console.WriteLine(temp.Model + " is already in the lot.");
// Удалить все
    nbLot.ClearAllNBooks();
```

```

    Console.WriteLine("You have {0} in the lot.\n", nbLot.NBookCount);
}
}
}

```

### **Содержание отчета**

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;
- 3) формулировки индивидуальных заданий и порядка их выполнения.

### **Контрольные вопросы**

1. Массивы каких трех видов могут использоваться в C#?
2. Каким образом можно описать массив в C#?
3. Перечислите основные свойства класса System.Array. Дайте их характеристику.
4. Охарактеризуйте методы класса System.Array.
5. Какой тип используется для описания строки постоянной длины в C#?
6. Каким образом объявляются объекты класса String?
7. Назначение метода String.
8. Какие операции выполняются в классе String?
9. Что представляет собой интерфейс в C#?
10. Перечислите основные интерфейсы, входящие в библиотеку базовых классов .Net. Дайте их характеристику

**Тема: Архитектура web-приложений ASP.NET. Разработка web-приложений на платформе .NET.**

### **ЛАБОРАТОРНАЯ РАБОТА 6**

#### **РАЗРАБОТКА WEB-ПРИЛОЖЕНИЙ С ПОМОЩЬЮ ASP.NET**

**Цель работы:** познакомить с основными этапами разработки Web-приложений на основе ASP.NET в среде Microsoft Visual Studio.NET; изучить структуру проекта ASP.NET Web Application.

**Формируемые компетенции или их части:**

ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий

## Теоретическая часть

ASP.NET – файл является текстовым файлом и может содержать коды HTML, XML и языков сценариев. Коды последних выполняются на Web-сервере. Файл ASP.NET имеет специальное расширение «.aspx».

Порядок работы ASP.NET – файла выглядит следующим образом:

- когда Web-браузер запрашивает файл ASP.NET, Web-сервер IIS перенаправляет запрос модулю ASP.NET на сервере;
- модуль ASP.NET читает файл построчно и выполняет, коды сценариев, содержащиеся в файле;
- Web-браузеру возвращается обратно файл ASP.NET, но уже в виде обычного HTML-документа.

Любая страница ASP.NET представлена классом, производным от класса System.Web.UI, который определяет свойства, методы и события, общие для всех страниц, предназначенных для обработки средой ASP.NET.

Таблица 6.1 Свойства объекта System.Web.UI

Свойство	Описание
Application	Возвращает объект HttpSessionState
Cache	Возвращает объект Cache, в котором хранятся данные приложения, в т. ч. и самой страницы
IsPostBack	Возвращает значение, определяющее, была ли страница загружена клиентом в первый раз, или загружена повторно в ответ на запрос клиента
Request	Возвращает объект HttpRequest, используемый для получения информации о входящем запросе HTTP
Response	Возвращает объект HttpResponse, используемые для формирования ответа сервера клиенту
Server	Возвращает объект HttpServerUtility
Session	Возвращает объект System.Web.SessionState.HttpSessionState, с помощью которого получается информация о текущем сеансе HTTP

Такое построение проекта позволяет хранить отдельно код представления для генерации HTML кода (в файле \*.aspx) от программной логики (в файле \*.aspx.cs), что во многих случаях существенно упрощает разработку сложных Web-приложений.

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт; устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц, среда программирования Visual Studio .Net.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### Задание

Создайте ASP.NET-приложение для вывода списка всех переменных окружения Web-сервера, добавив в него код:

```
< %
// метод AllKeys формирует список всех ключей массива
string[] rkeys = Request.ServerVariables.AllKeys;
// создание переменной типа StringBuilder (т.е. изменяемой строки)
StringBuilder output = new StringBuilder();
// перебор всех элементов массива rkeys и формирование списка значе-
ний
foreach (string rkey in rkeys)
{
```

```

        output.Append(rkey + "=" + Request.ServerVariables[rkey] + "<br>");
    }
// формирования ответа сервера
    Response.Write(output);
%>

```

### Указания по порядку выполнения работы

#### *Создание нового проекта в среде Microsoft Visual Studio с использованием шаблона ASP.NET Web Application*

Каждое из Web-приложений для IIS должно размещаться в своем виртуальном каталоге, которому соответствует физический каталог на диске. Для создания виртуального каталога необходимо:

1. Открыть оснастку Microsoft Internet Information Services (Пуск → Панель управления → Администрирование → Internet Information Services).
2. Раскрыть ветку «Web-узлы» и, перейдя в «Web-узел по умолчанию», создать новый виртуальный каталог.
3. В свойствах созданного виртуального каталога выбрать закладку «Документы» и добавить в список документов, используемых по умолчанию, документ Default.aspx.

После завершения создания проекта, он будет содержать файлы Default.aspx, Default.aspx.cs и Default.asp.designer.cs.

Первый из них будет содержать примерно следующий код:

```

<% @ Page Language="C#" AutoEventWireup="true" CodeBehind="
    Default.aspx.cs" Inherits="ASPNETHello._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>

```

```

<form id="form1" runat="server">
  <div>
  </div>
</form>
</body>
</html>

```

Из этого кода видно, что, во-первых, для создания кода *HTML*, возвращаемого браузеру, будет использован язык *C#*. Во-вторых, код *C#* содержится в отдельном файле, который будет выполняться на Web-сервере. И, наконец, атрибут *Inherits* указывает на имя класса, определенного в *CodeBehind*.

Важным новшеством в *ASP.NET* является атрибут `runat = "server"`, размещенный в тэге `<form>`. Он означает, что данный элемент должен быть обработан средой выполнения *ASP.NET*.

Теперь между тэгами `<div>` и `</div>` можно вставить код:

```

<h1>Hello!</h1>
<h1>I am</h1>
<%=Request.ServerVariables["HTTP_USER_AGENT"] %>>

```

В данном примере было использовано свойство *Request* объекта, производного от *Page*, для получения значения переменной окружения `HTTP_USER_AGENT`.

Код *C#*, который содержится в файле `Default.aspx.cs` может выглядеть примерно следующим образом:

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;

```

```

using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
namespace ASPNETHello
{
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
}
}

```

Здесь содержится описание метода Page\_Load, который вызывается при загрузке Web-страницы Default.aspx из виртуального каталога для данного проекта. В данном примере метод содержит пустое тело.

После компиляции проекта (опция меню «Build» или < Shift+F6 >) и его выполнения (< Ctrl+F5 >) можно будет увидеть в браузере страницу примерно следующего вида:

Hello!

I am

Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322;  
.NET CLR 2.0.50727; .NET CLR 3.0.04506.30; InfoPath.2;  
.NET CLR 3.0.04506.648; .NET CLR 3.5.21022)

### Содержание отчета

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;
- 3) формулировки индивидуальных заданий и порядка их выполнения.

### Контрольные вопросы

1. Что представляет собой ASP.Net-файл?
2. Алгоритм работы ASP.Net-файла.
3. С помощью какого класса описывается страница в ASP.Net?
4. Охарактеризуйте свойства объектов System.Web.UI.

## **Тема: Клиент-серверные технологии Web. Протокол HTTP ЛАБОРАТОРНАЯ РАБОТА 7**

### **СЕРВЕРНЫЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ ASP.NET**

**Цель работы:** познакомить со средствами автоматизации разработки Web-приложений в ASP.NET в виде серверных элементов управления WebForm.

#### **Формируемые компетенции или их части:**

1. ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий

#### **Теоретическая часть**

Важной особенностью ASP.NET-файлов является использование серверных элементов управления на Web-странице (элементы WebForm), которые являются фактически тэгами, понятными веб-серверу. Эти элементы определены в пространстве имен System.Web.UI.WebControls.

Принято выделять три типа серверных элементов управления:

- серверные элементы управления HTML – обычные HTML тэги;
- элементы управления Web-сервера – новые тэги ASP.NET;
- серверные элементы управления для проверки данных (валидации) – применяются для валидации входных данных от клиентского приложения (обычно Web-браузера).

Преимущества от использования таких элементов при разработке Web-приложений:

1. Сокращается количество кода, написанного вручную (что особенно заметно для сложных элементов документа). Элемент просто «перетаскивается» из панели инструментов, после чего выполняется настройка его параметров в специальном окне. При этом все изменения автоматически заносятся непосредственно в \*.aspx файл.

2. С программной точки зрения каждому из этих элементов управления соответствует определенный класс в библиотеке базовых классов .NET, что позволяет писать для них такой же код как и для любых других классов.

3. Для любого элемента управления WebForm определен набор событий, обрабатываемых на Web-сервере.

4. Для любого элемента управления WebForm предоставляется возможность для проверки ввода данных пользователем.

### *Серверные элементы управления HTML*

По умолчанию такие элементы управления в ASP.NET-файлах рассматриваются как текст. Для их программирования требуется добавление атрибута `runat="server"` в соответствующий HTML-элемент. Кроме того, все серверные элементы управления HTML должны быть размещены внутри области действия тэга `<form>`, также имеющего атрибут `runat="server"`.

Таблица 7.1

<b>Серверный элемент управления HTML</b>	<b>Описание</b>
HtmlAnchor	Управление HTML элементом <code>&lt;a&gt;</code>
HtmlButton	Управление HTML элементом <code>&lt;button&gt;</code>
HtmlForm	Управление HTML элементом <code>&lt;form&gt;</code>
HtmlGeneric	Управляет HTML элементами не описываемыми как элементы управления HTML, например, <code>&lt;body&gt;</code> , <code>&lt;div&gt;</code> , <code>&lt;span&gt;</code> и др.
HtmlImage	Управление HTML элементом <code>&lt;img&gt;</code>
HtmlInputButton	Управление HTML элементами <code>&lt;input type="button"&gt;</code> , <code>&lt;input type="submit"&gt;</code> и <code>&lt;input type="reset"&gt;</code>
HtmlInputCheckBox	Управление HTML элементом <code>&lt;input type="checkbox"&gt;</code>
HtmlInputFile	Управление HTML элементом <code>&lt;input type="file"&gt;</code>
HtmlInputHidden	Управление HTML элементом <code>&lt;input type="hidden"&gt;</code>
HtmlInputImage	Управление HTML элементом <code>&lt;input type="image"&gt;</code>
HtmlInputRadioButton	Управление HTML элементом <code>&lt;input type="radio"&gt;</code>
HtmlInputText	Управление HTML элементами <code>&lt;input type="text"&gt;</code> и <code>&lt;input type="password"&gt;</code>
HtmlSelect	Управление HTML элементом <code>&lt;select&gt;</code>
HtmlTable	Управление HTML элементом <code>&lt;table&gt;</code>
HtmlTableCell	Управление HTML элементами <code>&lt;td&gt;</code> и <code>&lt;th&gt;</code>
HtmlTableRow	Управление HTML элементом <code>&lt;tr&gt;</code>
HtmlTextArea	Управление HTML элементом <code>&lt;textarea&gt;</code>

### Элементы управления Web-сервера

Подобно серверным элементам управления HTML элементы данного типа также создаются на веб-сервере и предполагают добавление атрибута `runat="server"`. Однако они могут не соответствовать конкретным элементам HTML, но представлять более сложные элементы.

Общий синтаксис для описания таких элементов:

```
<asp:тип_элемента id="идентификатор" runat="server"/>
```

Таблица 7.2

Элемент управления Web-сервера	Описание
AdRotator	Банерная рулетка
Button	Отображение кнопки
Calendar	Отображение календаря
CalendarDay	Элемент выбора дня календаря
CheckBox	Отображение флажка
CheckBoxList	Группа флажков
DataGrid	Отображение полей источника данных
DataList	Отображение элементов из источника данных с помощью шаблонов
DropDownList	Выпадающий список
HyperLink	Гиперссылка
Image	Изображение
ImageButton	Кнопка в виде изображения
Label	Отображение статического содержимого, доступного для программирования, и с возможностью задания стиля
LinkButton	Кнопка с гиперссылкой
ListBox	Выпадающий список с единичным или множественным выделением
ListItem	Элемент списка
Literal	Отображение статического содержимого, доступного для программирования, но без возможности задания стиля
Panel	Контейнер для других элементов управления

Placeholder	Место для добавления элементов управления программным способом
RadioButton	Радио-кнопка
RadioButtonList	Группа радио-кнопок
BulletedList	Маркерный список
Repeater	Повторяемый список элементов
Style	Задание стиля элемента управления
Table	Таблица
TableCell	Ячейка таблицы
TableRow	Строка таблицы
TextBox	Поле для ввода текста
Xml	Отображение XML файла или результата XSLT преобразования

*Серверные элементы управления для проверки данных (валидации)*

Элементы управления данного типа применяются для проверки вводимых данных. Имеют следующий *синтаксис*:

```
<asp:тип_элемента id="идентификатор" runat="server" />
```

Таблица 7.3

<b>Элемент управления для проверки данных</b>	<b>Описание</b>
CompareValidator	Сравнивает значение, введенное в один элемент управления со значением, введенным в другой элемент, либо с фиксированным значением
CustomValidator	Позволяет задавать пользовательский метод проверки вводимых значений
RangeValidator	Проверяет, что значение, введенное пользователем, находится между двумя величинами
RegularExpressionValidator	Проверяет введенное значение на соответствие указанному шаблону
RequiredFieldValidator	Проверяет обязательное наличие введенного значения
ValidationSummary	Отображает отчет обо всех ошибках проверки значений, произошедших на веб-странице

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт; устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц, среда программирования Visual Studio .Net.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### **Задание**

Создайте Web-приложение ASP.NET, поддерживающее ввод на странице международного телефонного номера пользователем и использующее серверный элемент управления `<asp:RegularExpressionValidator>`.

### **Указания по порядку выполнения работы**

#### *1. Серверные элементы управления HTML*

1. Создайте новый проект по шаблону ASP.NET Web Application.
2. На странице Default.aspx добавьте в форму стандартные HTML элементы ввода данных.

Код формы должен выглядеть следующим образом:

```
<form id="form1" runat="server">
```

```

<div>
    Ваша оценка:<br><br>
    <input id="r1" type="radio" title="Отлично" value="Отлично"
runat="server"/>Отлично<br />
    <input id="r2" type="radio" value="Хорошо" runat="server" />Хорошо<br
/>
    <input id="r3" type="radio" value="Удовлетворительно" runat="server"
/>Удовлетворительно<br />
    <input id="r4" type="radio" value="Неудовлетворительно" runat="server"
/>Неудовлетворительно<br />
    <p id="ans" runat="server"></p>
    <br>
    <input id="Submit1" type="submit" value="Отправить" OnServer-
Click="submit" runat="server" /><br />
</div>
</form>

```

Обработчик события OnServerClick необходимо описать в файле Default.aspx.cs. Код метода submit:

```

protected void submit(object sender, System.EventArgs e)
{
    if (r1.Checked == true)
        ans.InnerHtml = "Вы оценили на отлично";
    else if (r2.Checked == true)
        ans.InnerHtml = "Вы оценили на хорошо";
    else if (r3.Checked == true)
        ans.InnerHtml = "Вы оценили на удовлетворительно";
    else if (r4.Checked == true)
        ans.InnerHtml = "Вы оценили на неудовлетворительно";
    else
        ans.InnerHtml = "Вы не выбрали оценку";
}

```

3. Выполните компиляцию проекта и запустите его на выполнение (либо вручную загрузите документ в браузере, указав URL соответствующего виртуального каталога).

### *Элементы управления Web-сервера*

Если для вновь созданного проекта типа *ASP.NET Web Application* в файле *Default.aspx* внутри формы (между тэгами `<form>` и `</form>`) вставить следующий код (или сделать это перетаскиванием нужных элементов из панели инструментов, раздел "Standard"):

```
<asp:ListBox id="LstBx" runat="server" width="100" height="80" AutoPostBack="True" OnSelectedIndexChanged="LBSelChanged" >
<asp:ListItem value="Sunday">Sunday</asp:ListItem>
<asp:ListItem value="Monday">Monday</asp:ListItem>
<asp:ListItem value="Tuesday">Tuesday</asp:ListItem>
<asp:ListItem value="Wednesday">Wednesday</asp:ListItem>
<asp:ListItem value="Thursday">Thursday</asp:ListItem>
<asp:ListItem value="Friday">Friday</asp:ListItem>
<asp:ListItem value="Saturday">Saturday</asp:ListItem>
</asp:ListBox>
<br/><br/>
<asp:Label id="Label1" runat="server"/>
```

и описать обработку событий `Page_Load` и `OnSelectedIndexChanged` в соответствующем `*.aspx.cs` файле:

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "You selected: " + Label1.Text;
}
protected void LBSelChanged(object sender, System.EventArgs e)
{
    if (LstBx.SelectedItem != null)
        Label1.Text = "You selected: " + LstBx.SelectedItem.Value;
    else
```

```
        Label1.Text = "You selected: ";  
    }  
}
```

В данном примере атрибут `AutoPostBack = "True"` указывает на необходимость немедленной обработки события на сервере.

*Серверные элементы управления для проверки данных (валидации)*

Следующий код показывает, каким образом выполняется проверка содержимого поля ввода формы:

```
<form id="form1" runat="server">  
    <asp:TextBox ID="TextBox1" runat="server" Width="123px"></asp:TextBox>  
    <br />  
    <asp:RegularExpressionValidator  
ID="RegularExpressionValidator1"  
runat="server"  
ControlToValidate="TextBox1"  
ErrorMessage="Not valid e-mail"  
ValidationExpression="\w+([-+.'\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*" >  
    </asp:RegularExpressionValidator>  
</form>
```

В данном случае атрибут `ControlToValidate` указывает на то, что контролируется содержимое элемента с идентификатором `TextBox1` (поля ввода текста), в случае несоответствия содержимого контролируемого поля с шаблоном, описанном в виде регулярного выражения в атрибуте `ValidationExpression`, выдается сообщение, указанное в атрибуте `ErrorMessage`.

Шаблон описывает допустимый формат адреса электронной почты.

Добавьте код в проект и проверьте работу приложения.

### **Содержание отчета**

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;

3) формулировки индивидуальных заданий и порядка их выполнения.

### **Контрольные вопросы**

1. Что является важной особенностью ASP.Net-файлов?
2. Перечислите основные типы серверных элементов управления.
3. Охарактеризуйте преимущества использования серверных управляющих элементов при разработке Web-приложений.
4. Дайте характеристику серверным элементам управления HTML.
5. Перечислите какие элементы управления используются на Web-сервере.
6. Охарактеризуйте серверные элементы управления для проверки данных (валидации).

### **Тема: Архитектура web-приложений ASP.NET. Разработка web-приложений на платформе .NET.**

#### **ЛАБОРАТОРНАЯ РАБОТА 8**

#### **РАБОТА С ИСТОЧНИКАМИ ДАННЫХ В ASP.NET**

**Цель работы:** изучение элементов WebForm, предназначенных для отображения на Web-странице данных, получаемых из источников данных.

#### **Формируемые компетенции или их части:**

1. ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий.

#### **Теоретическая часть**

В ASP.NET используются два элемента управления WebForm для управления отображением данных, получаемых из источника данных:

- DataGrid – элемент управления, отображающий содержимое объекта ADO.NET DataSet в виде таблицы;

- DataList – элемент управления для выбора значений, заполняемых из источника данных.

Если необходимо отобразить данные, полученные по запросу пользова-

теля из источника данных, в виде таблицы на Web-странице, то ASP.NET предоставляет в распоряжение Web-программиста удобный элемент управления DataGrid, который был введен в ASP.NET 1.x. но теперь его функции перекрываются GridView. Элемент управления GridView может не только показывать данные, но и сортировать, выбирать, редактировать их. Если этой функциональности недостаточно, ее можно расширить, написав собственные обработчики событий.

Элементы, которые могут быть связаны с источниками данных, многообразны, например, DropDownList, ListBox, CheckBoxList, RadioButtonList, BulletedList.

### *Элемент GridView (DataGrid)*

Элемент управления GridView – главный элемент управления для представления информации из баз данных в ASP.NET. Он является "наследником" элемента управления DataGrid, который использовался в предыдущих версиях ASP.NET. Основное назначение этого элемента управления – представление пользователю информации в табличном виде из источника данных с возможностью фильтрации, сортировки и редактирования.

Проще всего настроить этот элемент управления при помощи встроенного мастера. При использовании мастера необходимо:

1. Выбрать тип источника данных. В свойствах источника данных (элемента управления DataSource) настроить параметры подключения к базе данных).
2. Сгенерировать или ввести код команд SELECT, INSERT, UPDATE и DELETE. При этом необязательно заполнять данные для всех команд.
3. Настроить дополнительные параметры объекта GridView, например, разрешить разбиение на страницы, фильтрацию, сортировку, выбрать оформление и т.п.

### *DataList*

По сравнению с элементом управления GridView элемент управления DataList предоставляет больше возможностей по настройке внешнего вида

данных, которые берутся из источника, однако требует существенно больше времени и усилий по настройке. Добавлять новые записи, также как и при использовании элемента управления GridView, средствами DataList невозможно.

Работа с этим элементом управления DataList (на закладке Default.aspx в режиме Design) выглядит следующим образом:

1. После перетаскивания элемента управления DataList в форму необходимо настроить для него источник данных (либо используя визуальный интерфейс – с помощью объекта DataSource, либо программно через свойство DataSource – DataSet).

2. Далее нужно перейти в режим редактирования шаблонов для записей. Для этого в контекстном меню выбрать EditItemTemplate.

3. Для Item Templates появятся четыре области:

- ItemTemplate – шаблон для отображения обычных элементов на форме;

- AlternatingItemTemplate – необязательный шаблон для отображения каждой второй записи. Обычно для него только чуть меняется фон по сравнению с ItemTemplate – чтобы удобнее было воспринимать длинные страницы;

- SelectedItemTemplate – шаблон для отображения выбранной в настоящий момент записи;

- EditItemTemplate – шаблон для отображения редактируемой в настоящий момент записи.

Эти свойства можно также определить и через раздел Properties в контекстном меню элемента управления DataList. Можно ограничиться также простым выбором одного из шаблонов автоформатирования.

4. Затем нужно заполнить требуемыми элементами управления (и просто кодом HTML) каждый из шаблонов. Для каждого поля, информацию о котором нужно предоставлять пользователям, необходимо создать свой элемент управления. Для всех шаблонов, кроме EditItemTemplate, для большин-

ства полей будут использоваться элементы управления Label (хотя возможны и другие варианты). Для EditItemTemplate чаще всего используются элементы управления TextBox. Чтобы связать элементы управления с полями в источнике данных, проще всего использовать ссылку EditDataBindings в мастере, который появляется для каждого из добавляемых элементов управления.

**Оборудование и материалы.** Для выполнения лабораторной работы необходим персональный компьютер со следующими характеристиками: процессор Intel с тактовой частотой 2000 МГц и выше; оперативная память – не менее 1024 Мбайт; свободное дисковое пространство – не менее 1,2 Гбайт; устройство для чтения компакт-дисков; монитор типа Super VGA (число цветов – 256) с диагональю не менее 17 ". **Программное обеспечение** – операционная система WINDOWS XP и выше, программы для просмотра Web-страниц, среда программирования Visual Studio .Net.

**Указания по технике безопасности.** Техника безопасности при выполнении лабораторной работы совпадает с общепринятой для работы с персональным компьютером. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения. В случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории (оператору, администратору). Соблюдать правила техники безопасности при работе с электрооборудованием. Не касаться электрических розеток металлическими предметами. Рабочее место пользователя персонального компьютера должно содержаться в чистоте. Не разрешается возле персонального компьютера принимать пищу, напитки.

### **Задание**

Создайте Web-приложение ASP.NET, отображающее на веб-странице содержимое таблицы Exam, содержащей поля: идентификатор записи, фамилия студента, название дисциплины, оценка. При разработке используйте элемент <asp:GridView> или <asp:DataList>.

## Указания по порядку выполнения работы

### Использование элемента GridView (DataGrid)

В приведенном ниже примере в качестве источника данных используется база данных в формате MS ACCESS, содержащая таблицу lesson с полями: id (тип - счетчик); Title (тип - текст); Teacher (тип - текст); Type (тип - текст).

1. Создайте файл в формате MS ACCESS, содержащий эту таблицу.
2. Создайте новый проект по шаблону ASP.NET Web Application.
3. Для страницы Default.aspx выберите режим отображения Split.
4. Из панели элементов управления Toolbox (отображается с помощью меню View → Toolbox) перетащите в форму элемент GridView (из группы Data).

5. Для элемента GridView настройте источник данных (Choose Data Source, Configure Data Source) и форматирование (можно выбрать один из шаблонов автоформатирования – Auto Format ).

Автоматически сгенерированный код представления (файл Default.aspx ) может выглядеть следующим образом:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
```

```
Inherits="ASPNETGridView2._Default" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head runat="server">
```

```
<title>Untitled Page</title>
```

```
</head>
```

```
<body>
```

```
<form id="form1" runat="server">
```

```
<div>
```

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
```

```

        BackColor="White" BorderColor="#E7E7FF" BorderStyle="None" BorderWidth="1px"
        CellPadding="3" DataKeyNames="id" DataSourceID="AccessDataSource1"
        GridLines="Horizontal">
        <FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
        <RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
        <Columns>
            <asp:BoundField DataField="id" HeaderText="id" InsertVisible="False"
                ReadOnly="True" SortExpression="id" />
            <asp:BoundField DataField="Title" HeaderText="Title" SortExpression="Title" />
            <asp:BoundField DataField="Teacher" HeaderText="Teacher"
                SortExpression="Teacher" />
            <asp:BoundField DataField="Type" HeaderText="Type" SortExpression="Type" />
        </Columns>
        <PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" HorizontalAlign="Right" />
        <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="#F7F7F7" />
        <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />
        <AlternatingRowStyle BackColor="#F7F7F7" />
    </asp:GridView>
    <asp:AccessDataSource ID="AccessDataSource1" runat="server"
        DataFile="D:\CSharp\ASPNETDataGrid\lessons.mdb"
        SelectCommand="SELECT * FROM [lesson]"></asp:AccessDataSource>
</div>
</form>
</body>

```

```
</html>
```

В данном примере использован элемент управления `<asp:DataGrid>`, имеющий идентификатор "GridView1".

Если источник данных не настроен автоматически (при описании кода представления):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:DataGrid ID="GridView1" runat="server" BackColor="White"
        BorderColor="White" BorderStyle="Ridge" BorderWidth="2px" CellPad-
        ding="3"
        CellSpacing="1" GridLines="None" >
        <FooterStyle BackColor="#C6C3C6" ForeColor="Black" />
        <SelectedItemStyle BackColor="#9471DE" Font-Bold="True" ForeCol-
        or="White" />
        <PagerStyle BackColor="#C6C3C6" ForeColor="Black" Horizonta-
        lAlign="Right" />
        <ItemStyle BackColor="#DEDFDE" ForeColor="Black" />
        <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeCol-
        or="#E7E7FF" />
      </asp:DataGrid>
    </div>
  </form>
</body>
</html>
```

то это можно сделать программно в соответствующем программном коде, например:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Data.OleDb;
namespace ASPNETDataGrid
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                // Подключение к источнику данных
                OleDbConnection cn = new OleDbConnection();
                cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;"
                +           @"data source =           D:\From-Nb-
D\Work\method\CSharp\ASPNETDataGrid\Lessons.mdb";
                cn.Open();
                // Формируется строка SQL запроса данных из источника
```

```

        string str = "SELECT * from lesson";
// Происходит соединение с базой данных
// с помощью управляемого провайдера OLE DB
        OleDbDataAdapter dAdapt = new OleDbDataAdapter(str, cn);
// Получение данных из источника
        DataSet myDS = new DataSet("lessons");
        dAdapt.Fill(myDS, "lesson");
// Заполнение таблицы данными
        GridView1.DataSource = myDS.Tables["lesson"].DefaultView;
        GridView1.DataBind();
        cn.Close();
    }
}
}
}

```

Следует обратить внимание на то, что в программе добавлено пространство имен:

```
using System.Data.OleDb;
```

Свойство объекта Page - IsPostBack показывает, была ли страница клиента загружена в первый раз или повторно в ответ на переданные клиентом данные.

### *Использование элемента DataList*

В следующем примере в качестве источника данных также будем использовать базу данных в формате MS ACCESS, содержащую таблицу lesson.

1. Создайте новый проект по шаблону ASP.NET Web Application.
2. Для страницы Default.aspx выберите режим отображения Split
3. Из панели элементов управления Toolbox (отображается с помощью меню View → Toolbox) перетащите в форму элемент DataList (из группы Data).
4. Для элемента DataList настройте источник данных (Choose Data

Source, Configure Data Source) и форматирование (можно выбрать один из шаблонов автоформатирования – Auto Format).

Получится следующий код представления:

```
<% @ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs"
Inherits="ASPNETGVView2._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:DataList ID="DataList1" runat="server" BackColor="White"
BorderColor="#999999" BorderStyle="Solid" BorderWidth="1px" Cell-
Padding="3"
DataKeyField="id" DataMember="DefaultView" Data-
SourceID="AccessDataSource1"
ForeColor="Black" GridLines="Vertical" style="margin-right: 0px"
Width="262px" >
<FooterStyle BackColor="#CCCCCC" />
<AlternatingItemStyle BackColor="#CCCCCC" />
<SelectedItemStyle BackColor="#000099" Font-Bold="True" ForeCol-
or="White" />
<HeaderStyle BackColor="Black" Font-Bold="True" ForeColor="White"
/>
%>' />
<ItemTemplate> id:
<asp:Label ID="idLabel" runat="server" Text='<%=# Eval("id") %>' />
```

<br /> Title:

```
<asp:Label ID="TitleLabel" runat="server" Text='<%# Eval("Title")
```

<br /> Teacher:

```
<asp:Label ID="TeacherLabel" runat="server" Text='<%#  
Eval("Teacher") %>' />
```

```
<br />
```

Type:

```
%>' />
```

```
<asp:Label ID="TypeLabel" runat="server" Text='<%# Eval("Type")
```

```
<br />
```

```
</ItemTemplate>
```

```
</asp:DataList>
```

```
<asp:AccessDataSource ID="AccessDataSource1" runat="server"
```

```
DataFile="D:\CSharp\ASPNETDataGrid\lessons.mdb"
```

```
SelectCommand="SELECT * FROM [lesson]">
```

```
</asp:AccessDataSource>
```

```
</div>
```

```
</form>
```

```
</body>
```

```
</html>
```

Альтернативный вариант – использование метода `GetDataItem` объекта `DataBinder` для заполнения элементов вместо `<asp:Label>`.

Ниже представлен соответствующий код представления:

```
<% @ Page Language="C#" AutoEventWireup="true"
```

```
CodeBehind="Default.aspx.cs" Inherits="ASPNETDataGrid._Default" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
```

```
<head id="Head2" runat="server">
```

```

<title>Lessons</title>
</head>
<body>
  <form id="form2" runat="server">
    <div>
      <h1>Lessons</h1>
    </div>
    <asp:DataList ID="DataList1" runat="server" BackColor="White"
      BorderColor="White" BorderStyle="Ridge" BorderWidth="2px" CellPad-
ding="3"
      CellSpacing="1">
      <FooterStyle BackColor="#C6C3C6" ForeColor="Black" />
      <ItemStyle BackColor="#DEDFDE" ForeColor="Black" />
      <SelectedItemStyle BackColor="#9471DE" Font-Bold="True" ForeCol-
or="White" />
      <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeCol-
or="#E7E7FF" />
      <ItemTemplate>
        <%# DataBinder.GetDataItem(Container)%>
      </ItemTemplate>
    </asp:DataList>
  </form>
</body>
</html>

```

Программный код:

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;

```

```

using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
namespace ASPNETDataGrid
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                ArrayList DWeek = new ArrayList();
                DWeek.Add("Sunday");
                DWeek.Add("Monday");
                DWeek.Add("Tuesday");
                DWeek.Add("Wednesday");
                DWeek.Add("Thursday");
                DWeek.Add("Friday");
                DWeek.Add("Saturday");
                DataList1.DataSource = DWeek;
                DataList1.DataBind();
            }
        }
    }
}

```

*Использование элемента ListBox (с заполнением данными из  
обычного массива)*

1. Создайте новый проект по шаблону ASP.NET Web Application.
2. Для страницы Default.aspx выберите режим отображения Split
3. Из панели элементов управления Toolbox (отображается с помощью меню View → Toolbox) перетащите в форму элемент ListBox (из группы

Data).

Получится следующий код представления:

```
<% @ Page Language="C#" AutoEventWireup="true"
CodeBehind="Default.aspx.cs" Inherits="ASPNETListBox._Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="Form1" runat="server">
<asp:ListBox ID="ListBox1" runat="server"></asp:ListBox>
</form>
</body>
</html>
```

и соответствующий программный код:

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
namespace ASPNETListBox
{
```

```

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
// Массив строк, который нужно вставить
        ArrayList DWeek = new ArrayList();
        DWeek.Add("Sunday");
        DWeek.Add("Monday");
        DWeek.Add("Tuesday");
        DWeek.Add("Wednesday");
        DWeek.Add("Thursday");
        DWeek.Add("Friday");
        DWeek.Add("Saturday");

// Связывание элемента управления с объектом DWeek
        ListBox1.DataSource = DWeek;
        ListBox1.DataBind();
    }}}

```

### **Содержание отчета**

Отчет по лабораторной работе должен состоять из:

- 1) названия лабораторной работы;
- 2) ответов на контрольные вопросы;
- 3) формулировки индивидуальных заданий и порядка их выполнения.

### **Контрольные вопросы**

1. Какие два элемента WebForm используются для управления отображением данных?
2. Дайте характеристику элементу управления GridView (DataGrid).
3. Алгоритм настройки элемента управления GridView.
4. Охарактеризуйте элемент управления DataList.
5. Перечислите основные этапы работы с элементом управления DataList.

## СПИСОК ЛИТЕРАТУРЫ

### *Основная литература*

1. Информационные Web-технологии / Ю. Громов, О.Г. Иванова, Н.Г. Шахов, В.Г. Однолько; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». - Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2014. - 96 с. : ил. - Библиогр. в кн. - ISBN 978-5-8265-1365-1 ; То же [Электронный ресурс]. - URL: <http://biblioclub.ru/index.php?page=book&id=277935>

2. Тузовский А.Ф. Проектирование и разработка web-приложений [Электронный ресурс]: учебное пособие/ Тузовский А.Ф.— Электрон. текстовые данные.— Томск: Томский политехнический университет, 2014.— 219 с.— Режим доступа: <http://www.iprbookshop.ru/34702>.— ЭБС «IPRbooks»

### *Дополнительная литература*

1. Информационные Web-технологии / Ю. Громов, О. Г. Иванова, Н. Г. Шахов, В. Г. Однолько; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». – Тамбов: Издательство ФГБОУ ВПО «ТГТУ», 2014. – 96 с. [Электронный ресурс]. Режим доступа: <http://biblioclub.ru/index.php?page=book&id=277935>

2. Сухов, К. HTML5 – путеводитель по технологии / К. Сухов. – М.: ДМК Пресс, 2014. – 352 с. [Электронный ресурс]. Режим доступа: <http://biblioclub.ru/index.php?page=book&id=260322>

3. Котов, О. М. Язык C#: краткое описание и введение в технологии программирования: учебное пособие / О. М. Котов; Министерство образования и науки Российской Федерации, Уральский федеральный университет имени первого Президента России Б. Н. Ельцина. – Екатеринбург: Издательство Уральского университета, 2014. – 209 с.

[Электронный ресурс]. Режим доступа:  
<http://biblioclub.ru/index.php?page=book&id=275809>

4. Громов, Ю. Ю. Основы Web-инжиниринга: разработка клиентских приложений: учебное пособие / Ю. Ю. Громов, О. Г. Иванова, С. В. Данилкин; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». – Тамбов: Издательство ФГБОУ ВПО «ТГТУ», 2012. – 240 с. [Электронный ресурс]. Режим доступа:  
<http://biblioclub.ru/index.php?page=book&id=277648>

5. Кузин, А. В. Базы данных: учеб. пособие для вузов / А.В. Кузин, С.В. Левонисова. – 5-е изд., испр. – Москва: Академия, 2012. – 314 с. (электронный каталог библиотеки СКФУ)

6. Джонсон, Г. Разработка клиентских веб-приложений на платформе Microsoft NET FRAMEWORK: учеб. курс Microsoft: пер. с англ. / Гленн Джонсон, Тони Нортроп. – М.: Русская Редакция; СПб.: Питер, 2007. – 768 с. (электронный каталог СКФУ)

7. Базы данных: учебник для вузов / [Хомоненко А. Д., Цыганков В. М., Мальцев М. Г]; под ред. Хомоненко А. Д. - 3-е изд., перераб. и доп. - СПб.: КОРОНА принт, 2003. – 672 с (электронный каталог библиотеки СКФУ).

8. Основы WEB-технологий: Курс лекций. Специальность "Интернет-технологии" / П. Б. Храмов, С. А. Брик, А. М. Русак, А. И. Сурин. - М.: Интернет-Университет Информационных Технологий, 2003. – 512 с (электронный каталог библиотеки СКФУ)

9. Шумаков, П. В. ADO.Net и создание приложений баз данных в среде Microsoft Visual .Net. Руководство разработчика с примерами на C# / П. В. Шумаков. – М.: Диалог- МИФИ, 2003. – 528 с (электронный каталог библиотеки СКФУ)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

## **Методические указания**

по организации самостоятельной работы  
по дисциплине «Современные технологии программирования»

Для студентов направления подготовки 09.03.02 Информационные системы и технологии  
направленность (профиль) Цифровые технологии химических производств

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Невинномысск 2025

Методические указания предназначены для студентов направления подготовки 09.03.02 Информационные системы и технологии и других технических специальностей. Они содержат рекомендации по организации самостоятельных работ студентов для дисциплины «Современные технологии программирования».

Методические указания разработаны в соответствии с требованиями ФГОС ВО в части содержания и уровня подготовки выпускников направления 09.03.02 Информационные системы и технологии

## СОДЕРЖАНИЕ

1 Пояснительная записка.....	5
2 Критерии оценки результатов самостоятельной работы .....	5
3 Рекомендации по выполнению различных видов самостоятельных работ .....	5
3.1 Рекомендации по подготовке сообщения по изучаемой теме .....	6
3.2 Рекомендации по подготовке доклада .....	7
3.3 Рекомендации по написанию реферата .....	8
3.4 Рекомендации по подготовке электронной презентации.....	10
3.5 Методика составления опорного конспекта.....	11
3.6 Методика составления обобщающей таблицы .....	12
3.7 Правила самостоятельной работы с литературой.....	13
3.8 Рекомендации по написанию отчета по практической работе.....	15
4 Учебно-методическое и информационное обеспечение .....	16

## Введение

Внеаудиторная самостоятельная работа – планируемая учебная, учебно-исследовательская работа студентов, выполняемая вне занятий по заданию и при управлении преподавателем, но без его непосредственного участия.

Самостоятельная работа проводится с целью:

- систематизации и закрепления полученных теоретических знаний и практических умений обучающихся;
- углубления и расширения теоретических знаний;
- формирования умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развития познавательных способностей и активности обучающихся: творческой инициативы, самостоятельности, ответственности, организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, совершенствованию и самоорганизации;
- формирования общих и профессиональных компетенций
- развитию исследовательских умений.

## 1 Пояснительная записка

Самостоятельная работа студента – вид учебной деятельности студента, требующий большой подготовительной деятельности преподавателя. Самостоятельная работа позволяет оптимально сочетать теоретическую и практическую составляющие обучения. Плановая организация этой работы позволяет оперативно обновлять содержание образования, создавая предпосылки для формирования общих компетенций и обеспечивая, таким образом, качество подготовки специалистов на конкурентоспособном уровне.

Методические рекомендации раскрывают у студентов формирование системы знаний, практических умений и объяснения уровня образованности и уровня подготовки студентов. Изучение программного материала должно способствовать формированию у студентов необходимых для профессиональной деятельности знаний и навыков.

## 2 Критерии оценки результатов самостоятельной работы

Критериями оценки результатов внеаудиторной самостоятельной работы обучающихся являются:

- уровень освоения учебного материала;
- уровень умения использовать теоретические знания при выполнении практических задач;
- уровень сформированности обще учебных умений;
- уровень умения активно использовать электронные образовательные ресурсы, находить требующуюся информацию, изучать ее и применять на практике;
- обоснованность и четкость изложения материала;
- оформление материала в соответствии с требованиями стандарта предприятия;
- уровень умения ориентироваться в потоке информации, выделять главное;
- уровень умения четко сформулировать проблему, предложив ее решение, критически оценить решение и его последствия;
- уровень умения определить, проанализировать альтернативные возможности, варианты действий;
- уровень умения сформулировать собственную позицию, оценку и аргументировать ее.

## 3 Рекомендации по выполнению различных видов самостоятельных работ

В процессе самостоятельной работы студент приобретает навыки самоорганизации, самоконтроля, самоуправления, саморефлексии и становится активным самостоятельным субъектом учебной деятельности.

Выполняя самостоятельную работу под контролем преподавателя **студент должен:**

- освоить минимум содержания, выносимый на самостоятельную работу студентов и предложенный преподавателем в соответствии с ФГОС ВПО по данному профессиональному модулю;
- планировать самостоятельную работу в соответствии с графиком самостоятельной работы, предложенным преподавателем;
- самостоятельную работу студент должен осуществлять в организационных формах, предусмотренных учебным планом и рабочей программой преподавателя;
- выполнять самостоятельную работу и отчитываться по ее результатам в соответствии с графиком представления результатов, видами и сроками отчетности по самостоятельной работе студентов;

**студент может:**

сверх предложенного преподавателем (при обосновании и согласовании с ним) и минимума обязательного содержания, определяемого ФГОС ВПО по предмету:

- самостоятельно определять уровень (глубину) проработки содержания материала;
- предлагать дополнительные темы и вопросы для самостоятельной проработки;

- в рамках общего графика выполнения самостоятельной работы предлагать обоснованный индивидуальный график выполнения и отчетности по результатам самостоятельной работы;
- предлагать свои варианты организационных форм самостоятельной работы;
- использовать для самостоятельной работы методические пособия, учебные пособия, разработки сверх предложенного преподавателем перечня;
- использовать не только контроль, но и самоконтроль результатов самостоятельной работы в соответствии с методами самоконтроля, предложенными преподавателем или выбранными самостоятельно.

Самостоятельная работа студентов должна оказывать важное влияние на формирование личности будущего специалиста, она планируется студентом самостоятельно. Каждый студент самостоятельно определяет режим своей работы и меру труда, затрачиваемого на овладение учебным содержанием по каждой дисциплине. Он выполняет внеаудиторную работу по личному индивидуальному плану, в зависимости от его подготовки, времени и других условий.

### **3.1 Рекомендации по подготовке сообщения по изучаемой теме**

Подготовка сообщения – это вид внеаудиторной самостоятельной работы по подготовке небольшого по объему устного сообщения для озвучивания на аудиторном занятии. Сообщаемая информация носит характер уточнения или обобщения, несет новизну, отражает современный взгляд по определенным проблемам.

Сообщение отличается от докладов и рефератов не только объемом информации, но и ее характером – сообщения дополняют изучаемый вопрос фактическими или статистическими материалами. Оформляется задание письменно, оно может включать элементы наглядности (иллюстрации, демонстрацию, презентацию).

Регламент времени на озвучивание сообщения – до 5 мин.

Для выполнения самостоятельной работы необходимо:

- собрать и изучить источники информации по теме;
- составить план сообщения;
- выделить основные понятия;
- ввести в текст дополнительные данные, характеризующие объект изучения;
- оформить текст письменно;
- сдать на контроль преподавателю и озвучить в установленный срок.

#### **Критерии оценки сообщения по изучаемой теме:**

1. Актуальность темы.
2. Соответствие содержания теме; глубина проработки материала.
3. Грамотность и полнота использования источников; наличие элементов наглядности.

Оценка «ОТЛИЧНО» ставится, если сообщение соответствует заданию; тема глубоко проработана с использованием достаточного количества источников информации; сообщение содержит дополнительную информацию, не освещенную в учебнике; сообщение сопровождается элементами наглядности; выдержан регламент времени на озвучивание сообщения.

Оценка «ХОРОШО» ставится, если сообщение соответствует заданию; тема глубоко проработана с использованием достаточного количества источников информации; сообщение содержит дополнительную информацию, не освещенную в учебнике; сообщение не сопровождается элементами наглядности; регламент времени на озвучивание сообщения превышен.

Оценка «УДОВЛЕТВОРИТЕЛЬНО» ставится, если сообщение не вполне соответствует заданию; тема проработана неглубоко; использован один источник информации; сообщение содержит мало дополнительной информации, не освещено в учебнике; сообщение не сопровождается элементами наглядности; регламент времени на озвучивание сообщения не выдержан.

### **3.2 Рекомендации по подготовке доклада**

Доклад, как вид самостоятельной работы в учебном процессе, способствует формированию навыков исследовательской работы, расширяет познавательные интересы, учит критически мыслить.

При написании доклада по заданной теме студент составляет план, подбирает основные источники.

В процессе работы с источниками систематизирует полученные сведения, делает выводы и обобщения.

К докладу по крупной теме могут привлекать несколько студентов, между которыми распределяются вопросы выступления.

#### **Выбор темы доклада**

Тематика доклада обычно определяется преподавателем, но в определении темы инициативу может проявить и студент.

Прежде чем выбрать тему доклада, автору необходимо выявить свой интерес, определить, над какой проблемой он хотел бы поработать, более глубоко ее изучить.

#### **Этапы работы над докладом**

1) Формулирование темы, причем она должна быть не только актуальной по своему значению, но и оригинальной, интересной по содержанию.

2) Подбор и изучение основных источников по теме (как правильно, при разработке доклада используется не менее 8-10 различных источников).

3) Составление списка использованных источников.

4) Обработка и систематизация информации.

5) Разработка плана доклада.

6) Написание доклада.

7) Публичное выступление с результатами исследования.

#### **Структура доклада:**

– титульный лист

– оглавление (в нем последовательно излагаются названия пунктов доклада, указываются страницы, с которых начинается каждый пункт);

– введение (формулирует суть исследуемой проблемы, обосновывается выбор темы, определяются ее значимость и актуальность, указываются цель и задачи доклада, дается характеристика используемой литературы);

– основная часть (каждый раздел ее, доказательно раскрывая отдельную проблему или одну из ее сторон, логически является продолжением предыдущего; в основной части могут быть представлены таблицы, графики, схемы);

– заключение (подводятся итоги или дается обобщенный вывод по теме доклада, предлагаются рекомендации);

– список использованных источников.

Введение – это вступительная часть научно-исследовательской работы. Автор должен приложить все усилия, чтобы в этом небольшом по объему разделе показать актуальность темы, раскрыть практическую значимость ее, определить цели и задачи эксперимента или его фрагмента.

Основная часть. В ней раскрывается содержание доклада.

Как правило, основная часть состоит из теоретического и практического разделов.

В теоретическом разделе раскрываются история и теория исследуемой проблемы, дается критический анализ литературы и показываются позиции автора.

В практическом разделе излагаются методы, ход, и результаты самостоятельно проведенного эксперимента или фрагмента.

В основной части могут быть также представлены схемы, диаграммы, таблицы, рисунки и т.д.

В заключении содержатся итоги работы, выводы, к которым пришел автор, и рекомендации. Заключение должно быть кратким, обязательным и соответствовать поставленным задачам.

Список использованных источников представляет собой перечень использованных книг, статей, фамилии авторов приводятся в алфавитном порядке, при этом все источники даются под общей нумерацией литературы. В исходных данных источника указываются фамилия и инициалы автора, название работы, место и год издания.

Приложение к докладу оформляются на отдельных листах, причем каждое должно иметь свой тематический заголовок и номер, который пишется в правом верхнем углу, например: «Приложение 1».

#### **Требования к оформлению доклада**

Доклад выполняется на листах писчей бумаги формата А4 в Microsoft Word, объемом 5-10 страниц текста. Размер шрифта – 14; интервал – 1,5; с нумерацией страниц сверху страницы посередине, абзацный отступ на расстоянии 3 см от левой границы поля. В тексте обязательны ссылки на первоисточники. Количество источников: не менее 5-8 различных источников. При оформлении работы соблюдаются поля:

- левое – 30 мм;
- правое – 10 мм;
- нижнее – 20 мм;
- верхнее – 20 мм.

Доклад должен быть выполнен грамотно, с соблюдением культуры изложения.

#### **Критерии оценки доклада**

- актуальность темы исследования;
- соответствие содержания теме;
- глубина проработки материала; правильность и полнота использования источников;
- соответствие оформления доклада стандартам.

По усмотрению преподавателя доклады могут быть представлены на семинарах, научно-практических конференциях, а также использоваться как зачетные работы по пройденным темам.

### **3.3 Рекомендации по написанию реферата**

Реферат – это научно-исследовательская работа, в которой на основе изучения первоисточников (статей, книг, других материалов) автор раскрывает суть исследуемой проблемы, приводит различные точки зрения на изучаемую проблему (идеи, решения, предложения и т.д.), а также собственные взгляды, обосновывая свое мнение

Реферат, как правило, должен содержать следующие структурные элементы:

1. титульный лист;
2. содержание;
3. введение;
4. основная часть;
5. заключение;
6. список использованных источников;
7. приложения (при необходимости).

Объем реферата – не менее 5 и не более 15 страниц. В реферате следует сделать ссылки на использованные источники. Они должны быть оформлены в соответствии с установленным стандартом.

Готовый реферат представляется преподавателю для проверки. При оценке реферата учитывается умение обучающегося работать с научной литературой, анализировать различные точки зрения по спорным вопросам, аргументировать свое мнение, навыки оформления ссылок, списка использованной литературы.

В содержании приводятся наименования структурных частей реферата, глав и параграфов его основной части с указанием номера страницы, с которой начинается соответствующая часть, глава, параграф.

Во введении дается общая характеристика реферата: обосновывается актуальность выбранной темы; определяется цель работы и задачи, подлежащие решению для её достижения;

описываются объект и предмет исследования, информационная база исследования, а также кратко характеризуется структура реферата по главам.

Основная часть должна содержать материал, необходимый для достижения поставленной цели и задач, решаемых в процессе выполнения реферата. Она включает 2-3 главы, каждая из которых, в свою очередь, делится на 2-3 параграфа. Содержание основной части должно точно соответствовать теме проекта и полностью её раскрывать. Главы и параграфы реферата должны раскрывать описание решения поставленных во введении задач. Поэтому заголовки глав и параграфов, как правило, должны соответствовать по своей сути формулировкам задач реферата. Заголовок "ОСНОВНАЯ ЧАСТЬ" в содержании реферата быть не должно.

Главы основной части реферата могут носить теоретический, методологический и аналитический характер.

Обязательным для реферата является логическая связь между главами и последовательное развитие основной темы на протяжении всей работы, самостоятельное изложение материала, аргументированность выводов. Также обязательным является наличие в основной части реферата ссылок на использованные источники.

Изложение необходимо вести от третьего лица («Автор полагает...») либо использовать безличные конструкции и неопределенно-личные предложения («На втором этапе исследуются следующие подходы...», «Проведенное исследование позволило доказать...» и т.п.).

В заключении логически последовательно излагаются выводы, к которым пришел студент в результате выполнения реферата. Заключение должно кратко характеризовать решение всех поставленных во введении задач и достижение цели реферата.

Список использованных источников является составной частью работы и отражает степень изученности рассматриваемой проблемы. Количество источников в списке определяется студентом самостоятельно, для реферата их рекомендуемое количество от 10 до 20. При этом в списке обязательно должны присутствовать источники, изданные в последние 3 года, а также ныне действующие нормативно-правовые акты, регулирующие отношения, рассматриваемые в реферате.

В приложения следует относить вспомогательный материал, который при включении в основную часть работы загромождает текст (таблицы вспомогательных данных, инструкции, методики, формы документов и т.п.).

### **Оформление реферата**

При выполнении внеаудиторной самостоятельной работы в виде реферата необходимо соблюдать следующие требования:

- на одной стороне листа белой бумаги формата А4;
- размер шрифта – 12; Times New Roman, цвет – черный;
- междустрочный интервал – одинарный;
- поля на странице – размер левого поля – 2 см, правого – 1 см, верхнего – 2см, нижнего – 2см;
- отформатировано по ширине листа;
- на первой странице необходимо изложить план (содержание) работы;
- в конце работы необходимо указать источники использованной литературы;
- нумерация страниц текста – сквозная.

Список использованных источников должен формироваться в алфавитном порядке по фамилии авторов. Литература обычно группируется в списке в такой последовательности:

1. законодательные и нормативно-методические документы и материалы;
2. специальная научная отечественная и зарубежная литература (монографии, учебники, научные статьи и т.п.);
3. статистические, инструктивные и отчетные материалы предприятий, организаций и учреждений.

Включенная в список литература нумеруется сплошным порядком от первого до последнего названия.

По каждому литературному источнику указывается: автор (или группа авторов), полное название книги или статьи, место и наименование издательства (для книг и брошюр), год издания; для журнальных статей указывается наименование журнала, год выпуска и номер. По сборникам

трудов (статей) указывается автор статьи, ее название и далее название книги (сборника) и ее выходные данные.

Приложения следует оформлять как продолжение реферата на его последующих страницах.

Каждое приложение должно начинаться с новой страницы. Вверху страницы справа указывается слово "Приложение" и его номер. Приложение должно иметь заголовок, который располагается по центру листа отдельной строкой и печатается прописными буквами.

Приложения следует нумеровать порядковой нумерацией арабскими цифрами.

На все приложения в тексте работы должны быть ссылки. Располагать приложения следует в порядке появления ссылок на них в тексте.

### **Критерии оценки реферата**

Срок сдачи готового реферата определяется утвержденным графиком.

В случае отрицательного заключения преподавателя студент обязан доработать или переработать реферат. Срок доработки реферата устанавливается руководителем с учетом сущности замечаний и объема необходимой доработки.

Реферат оценивается по системе:

Оценка "ОТЛИЧНО" выставляется за реферат, который носит исследовательский характер, содержит грамотно изложенный материал, с соответствующими обоснованными выводами.

Оценка "ХОРОШО" выставляется за грамотно выполненный во всех отношениях реферат при наличии небольших недочетов в его содержании или оформлении.

Оценка "УДОВЛЕТВОРИТЕЛЬНО" выставляется за реферат, который удовлетворяет всем предъявляемым требованиям, но отличается поверхностностью, в нем просматривается непоследовательность изложения материала, представлены необоснованные выводы.

Оценка "НЕУДОВЛЕТВОРИТЕЛЬНО" выставляется за реферат, который не носит исследовательского характера, не содержит анализа источников и подходов по выбранной теме, выводы носят декларативный характер.

### **3.4 Рекомендации по подготовке электронной презентации**

Создание материалов-презентаций – это вид самостоятельной работы обучающихся по созданию наглядных информационных пособий, выполненных с помощью мультимедийной компьютерной программы PowerPoint.

Презентация как документ представляет собой последовательность сменяющих друг друга слайдов – то есть электронных страничек, занимающих весь экран монитора (без присутствия панелей программы). Чаще всего демонстрация презентации проецируется на большом экране, реже – раздается собравшимся как печатный материал. Количество слайдов адекватно содержанию и продолжительности выступления (например, для 5-минутного выступления рекомендуется использовать не более 10 слайдов).

Этапы создания презентации:

- планирование презентации (выбор темы; определение аудитории; определение цели);
- подготовка содержания презентации (структурно-семантический анализ темы презентации; поиск и аналитико-синтетическая переработка источников информации; составление плана презентации на основе имеющихся источников; написание исходного текста для презентации; разделение исходного текста на порции – по кадрам (экранам, слайдам), определение их последовательности; определение состава каждого кадра (экрана, слайда), включая изображения: рисунок, фото, таблица, диаграмма, схема; тексты: заголовок слайда, перечень вопросов, дефиниция, тезис, лозунг (слоган) и т. п.; определение содержания устного комментария к каждому слайду;
- техническая реализация презентации (использование возможностей программы PowerPoint (или иной программы) для подготовки мультимедийной презентации; использование мультимедийных эффектов;
- выбор дизайна презентации;
- репетиция презентации (проверка синхронности устного текста и демонстрируемых слайдов; обеспечение соответствия объема презентации отведенному на нее времени, проверка соответствия презентации требованиям устного публичного выступления).

В ходе создания презентации необходимо руководствоваться следующими требованиями.

Правила шрифтового оформления:

1. Шрифты с засечками читаются легче, чем гротески (шрифты без засечек);
2. Для основного текста не рекомендуется использовать прописные буквы.
3. Шрифтовой контраст можно создать посредством: размера шрифта, толщины шрифта,

начертания, формы, направления и цвета.

Правила выбора цветовой гаммы.

1. Цветовая гамма должна состоять не более чем из двух-трех цветов.
2. Существуют не сочетаемые комбинации цветов.
3. Черный цвет имеет негативный (мрачный) подтекст.
4. Белый текст на черном фоне читается плохо (инверсия плохо читается).

Правила общей композиции.

1. На полосе не должно быть больше семи значимых объектов, так как человек не в состоянии запомнить за один раз более семи пунктов чего-либо.

2. Логотип на полосе должен располагаться справа внизу (слева наверху и т. д.).

3. Логотип должен быть простой и лаконичной формы.

4. Дизайн должен быть простым, а текст – коротким.

5. Изображения домашних животных, детей, женщин и т.д. являются положительными образами.

6. Крупные объекты в составе любой композиции портят общее впечатление. Огромные буквы в заголовках, кнопки навигации высотой в 40 пикселей, верстка в одну колонку шириной в 600 точек, разделитель одного цвета, растянутый на весь экран – все это придает дизайну непрофессиональный вид.

**Критерии оценки электронной презентации**

1. Соответствие содержания теме.

2. Правильная структурированность информации.

3. Наличие логической связи изложенной информации.

4. Эстетичность оформления и соответствие его теме, соответствие требованиям оформления презентаций.

5. Работа представлена в срок.

Оценка «ОТЛИЧНО» ставится, если презентация соответствует заданию; тема глубоко проработана с использованием достаточного количества источников информации; презентация содержит дополнительную информацию, не освещенную в учебнике; оформление презентации эстетично и соответствует теме, соответствие требованиям оформления презентаций; эффекты анимации в презентации использованы дозированно, не отвлекая и не раздражая слушателей.

Оценка «ХОРОШО» ставится, если выдержаны перечисленные выше критерии, допущены незначительные нарушения в стилистике оформления, недочеты в структуре.

Оценка «УДОВЛЕТВОРИТЕЛЬНО» ставится, если тема презентации проработана неглубоко с использованием одного источника информации; нарушены требования оформления, структура проработана слабо.

### **3.5 Методика составления опорного конспекта**

Опорный конспект – это развернутый план Вашего предстоящего ответа на теоретический вопрос. Он призван помочь Вам последовательно изложить тему, а преподавателю – лучше понимать Вас и следить за логикой Вашего ответа.

Правильно составленный опорный конспект должен содержать все то, что в процессе ответа Вы намереваетесь рассказать. Это могут быть чертежи, графики, формулы (если требуется, с выводом), формулировки основных законов, определения.

**Основные требования к форме записи опорного конспекта:**

1. Лаконичность. Опорный конспект должен быть минимальным, чтобы его можно было воспроизвести за 6 – 8 минут. По объему он должен составлять примерно один полный лист.

2. Структурность. Весь материал должен располагаться малыми логическими блоками, т.е. должен содержать несколько отдельных пунктов, обозначенных номерами или строчными пробелами.

3. Акцентирование. Для лучшего запоминания основного смысла Опорного конспекта, главную идею Опорного конспекта выделяют рамками различных цветов, различным шрифтом, различным расположением слов (по вертикали, по диагонали).

4. Унификация. При составлении Опорный конспект используются определённые аббревиатуры и условные знаки, часто повторяющиеся в курсе данного предмета

5. Автономия. Каждый малый блок (абзац), наряду с логической связью с остальными, должен выражать законченную мысль, должен быть аккуратно оформлен (иметь привлекательный вид).

6. Оригинальность. Опорный конспект должен быть оригинален по форме, структуре, графическому исполнению, благодаря чему, он лучше сохраняется в памяти. ОК должен быть наглядным и понятным не только Вам, но и преподавателю.

7. Взаимосвязь. Текст Опорного конспекта должен быть взаимосвязан с текстом учебника, что так же влияет на усвоение материала.

### **Примерный порядок составления опорного конспекта:**

1. Первичное ознакомление с материалом изучаемой темы по тексту учебника, картам, дополнительной литературе.

2. Выделение главного в изучаемом материале, составление обычных кратких записей.

3. Подбор к данному тексту опорных сигналов в виде отдельных слов, определённых знаков, графиков, рисунков.

4. Продумывание схематического способа кодирования знаний, использование различного шрифта и т.д.

5. Составление опорного конспекта.

### **Критерии оценки опорного конспекта:**

1. Соответствие конспекта содержанию темы;

2. Правильная структурированность информации;

3. Наличие логической связи изложенной информации; соответствие оформления требованиям; аккуратность и грамотность изложения;

4. Работа сдана в срок.

Оценка «ОТЛИЧНО» ставится, если содержание опорного конспекта соответствует теме, информация правильно структурирована, изложена грамотно, лаконично и логически связано, с использованием схем и сокращений; работа сдана в установленный срок

Оценка «ХОРОШО» ставится, если содержание опорного конспекта соответствует теме, информация правильно структурирована, изложена лаконично и логически связано, с использованием схем и сокращений; работа сдана в установленный срок; не использованы опорные сигналы, допущены 1-2 незначительные ошибки и нарушения в оформлении

Оценка «УДОВЛЕТВОРИТЕЛЬНО» ставится, если содержание опорного конспекта соответствует теме, информация структурирована нерационально, изложена неаккуратно, без использования схем и сокращений; работа сдана не в установленный срок; не использованы опорные сигналы, допущены нарушения в оформлении.

### **3.6 Методика составления обобщающей таблицы**

Составление сводной (обобщающей) таблицы по теме – это вид самостоятельной работы обучающегося по систематизации объемной информации, которая сводится (обобщается) в рамки таблицы. Формирование структуры таблицы отражает склонность обучающегося к систематизации материала и развивает его умения по структурированию информации. Краткость изложения информации характеризует способность к ее свертыванию. В рамках таблицы наглядно отображаются как разделы одной темы, так и разделы разных тем. Такие таблицы создаются как помощь в изучении большого объема информации, желая придать ему оптимальную форму для

запоминания. Задание чаще всего носит обязательный характер, а его качество оценивается по качеству знаний в процессе контроля.

Затраты времени на составление сводной таблицы зависят от объема информации, сложности ее структурирования и определяется преподавателем.

**Порядок работы:**

- изучить информацию по теме;
- выбрать оптимальную форму таблицы;
- информацию представить в сжатом виде и заполнить ею основные графы таблицы;
- пользуясь готовой таблицей, эффективно подготовиться к контролю по заданной теме;

**Критерии оценки:**

- соответствие содержания теме;
- логичность структуры таблицы;
- правильный отбор информации;
- наличие обобщающего (систематизирующего, структурирующего, сравнительного) характера изложения информации;
- соответствие оформления требованиям; работа сдана в срок.

Оценка «ОТЛИЧНО» ставится, если содержание таблицы соответствует теме; тема глубоко проработана с использованием достаточного количества источников информации; таблица правильно структурирована; таблица аккуратно оформлена; работа сдана в срок.

Оценка «ХОРОШО» ставится, если выдержаны перечисленные выше критерии, допущены незначительные недочеты; работа сдана в срок.

Оценка «УДОВЛЕТВОРИТЕЛЬНО» ставится, если тема проработана неглубоко, структура таблицы проработана слабо; таблица содержит много лишней информации или наоборот не содержит всей необходимой информации; таблица оформлена неаккуратно; работа сдана не в срок.

### **3.7 Правила самостоятельной работы с литературой**

Самостоятельная работа с учебниками и книгами – это важнейшее условие формирования у себя научного способа познания. Основные советы здесь можно свести к следующим:

1. Составить перечень книг, с которыми Вам следует познакомиться.

2. Сам такой перечень должен быть систематизированным (что необходимо для семинаров, что для экзаменов, что пригодится для написания курсовых и дипломных работ, а что Вас интересует за рамками официальной учебной деятельности, то есть что может расширить Вашу общую культуру...).

3. Обязательно выписывать все выходные данные по каждой книге (при написании курсовых и дипломных работ это позволит очень сэкономить время).

4. При составлении перечней литературы следует посоветоваться с преподавателями (или даже с более подготовленными и эрудированными сокурсниками), которые помогут Вам лучше сориентироваться, на что стоит обратить большее внимание, а на что вообще не стоит тратить время...

5. Естественно, все прочитанные книги, учебники и статьи следует конспектировать, но это не означает, что надо конспектировать «все подряд»: можно выписывать кратко основные идеи автора и иногда приводить наиболее яркие и показательные цитаты (с указанием страниц).

6. Если книга – Ваша собственная, то допускается делать на полях книги краткие пометки или же в конце книги, на пустых страницах просто сделать свой «предметный указатель», где отмечаются наиболее интересные для Вас мысли и обязательно указываются страницы в тексте автора (это очень хороший совет, позволяющий экономить время и быстро находить «избранные» места в самых разных книгах).

7. Если Вы раньше мало работали с научной литературой, то следует выработать в себе способность «воспринимать» сложные тексты; для этого лучший прием – научиться «читать медленно», когда Вам понятно каждое прочитанное слово (а если слово незнакомое, то либо с помощью словаря, либо с помощью преподавателя обязательно его узнать); опыт показывает, что после этого студент быстро и качественно прорабатывает книги.

8. Есть еще один эффективный способ оптимизировать знакомство с научной литературой – следует увлечься какой-то идеей и все книги просматривать с точки зрения данной идеи. В этом случае студент (или молодой ученый) будет как бы искать аргументы «за» или «против» интересующей его идеи, и одновременно он будет как бы общаться с авторами этих книг по поводу своих идей и размышлений...

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того насколько осознанно читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Для студентов основным видом чтения учебной литературы является изучающее чтение, которое предполагает доскональное освоение материала; в ходе такого чтения готовность принять изложенную информацию, реализуется установка на предельно полное понимание материала; именно оно позволяет в работе с учебной литературой накапливать знания в различных областях. Вот почему именно этот вид чтения в рамках учебной деятельности должен быть освоен в первую очередь. Кроме того, при овладении данным видом чтения формируются основные приемы, повышающие эффективность работы с научным текстом.

#### ***Основные виды систематизированной записи прочитанного:***

1. Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;
2. Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;
3. Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;
4. Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;
5. Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

#### **Критерии оценки устного ответа студента:**

При оценке устных ответов студентов учитываются следующие критерии:

1. Знание основных процессов изучаемой предметной области, глубина и полнота раскрытия вопроса.
2. Владение терминологическим аппаратом и использование его при ответе.
3. Умение объяснить сущность явлений, событий, процессов, делать выводы и обобщения, давать аргументированные ответы.
4. Умение делать анализ рекламного продукта по предложенной схеме.
5. Владение монологической речью, логичность и последовательность ответа, умение отвечать на поставленные вопросы, выражать свое мнение по обсуждаемой проблеме.

Оценкой "ОТЛИЧНО" оценивается ответ, который показывает прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа.

Оценкой "ХОРОШО" оценивается ответ, обнаруживающий прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа. Однако допускается одна - две неточности в ответе.

Оценкой "УДОВЛЕТВОРИТЕЛЬНО" оценивается ответ, свидетельствующий в основном о знании процессов изучаемой предметной области, отличающийся недостаточной глубиной и

полнотой раскрытия темы; знанием основных вопросов теории; слабо сформированными навыками анализа явлений, процессов, недостаточным умением давать аргументированные ответы и приводить примеры; недостаточно свободным владением монологической речью, логичностью и последовательностью ответа. Допускается несколько ошибок в содержании ответа.

Оценкой "НЕУДОВЛЕТВОРИТЕЛЬНО" оценивается ответ, обнаруживающий незнание процессов изучаемой предметной области, отличающийся неглубоким раскрытием темы; незнанием основных вопросов теории, несформированными навыками анализа явлений, процессов; неумением давать аргументированные ответы, слабым владением монологической речью, отсутствием логичности и последовательности. Допускаются серьезные ошибки в содержании ответа.

### **3.8 Рекомендации по написанию отчета по практической работе**

Отчет по практической работе оформляют с использованием средств, которые предоставляются текстовым процессором MS Word и распечатывают на принтере с хорошим качеством печати.

#### **Оформление текста отчета по практической работе**

Текст должен располагаться на одной стороне листа бумаги формата А4, иметь книжную ориентацию для основного текста, и альбомную, если это необходимо, – для размещения схем, рисунков, таблиц и т.п.

Для страниц с книжной ориентацией рекомендуется устанавливать следующие размеры полей:

- верхнее – 2 см,
- нижнее – 2 см,
- левое – 3 см,
- правое – 1 см.

Для страниц с альбомной ориентацией рекомендуется устанавливать следующие размеры полей:

- верхнее – 3 см,
- нижнее – 1 см,
- левое – 2 см,
- правое – 2 см.

Для ввода (и форматирования) текста рекомендуется использовать:

- шрифт – Times New Roman,
- размер – 14 (12) пт,
- межстрочный интервал – одинарный,
- способ выравнивания – по ширине для основного текста (для заголовков, списков и других элементов текста можно выбирать другие способы выравнивания, например, заголовки можно размещать по центру),
- начертание – обычное,
- отступ первой строки (абзацный отступ) – 1,25 см.

Для выделения заголовков, ключевых понятий допускается использование других способов начертания (курсив, полужирное).

В тексте следует использовать автоматическую расстановку переносов.

Кавычки в тексте оформляются единообразно (либо « », либо “ ”).

#### **Списки**

Нумерованные списки оформляются следующим образом:

1. ....
2. ....
3. ....

Маркированные списки оформляются следующим образом:

- ....
- ....
- ....

### **Таблицы и иллюстрации**

В отчете по практической работе можно использовать таблицы, которые помогают систематизировать, структурировать и наглядно представлять материалы.

Ссылка на таблицу в тексте обязательна. Таблицу следует располагать в тексте лишь после её упоминания.

Иллюстрации (чертежи, графики, схемы, рисунки) также следует располагать в тексте после их первого упоминания, и на них обязательно должны быть ссылки.

Нумерация рисунков (таблиц) может быть сквозной по всей работе или осуществляться в пределах раздела, например, Рисунок 1 или Рисунок 1.1.

Ссылки на таблицы, рисунки, приложения заключаются в круглые скобки.

### **Нумерация страниц**

В отчете по практической работе осуществляется сквозная нумерация страниц, начиная с первого листа.

Порядковый номер страницы следует ставить в правом нижнем углу поля страницы. Страницы работы следует нумеровать арабскими цифрами.

Отчет по практической работе должен содержать: название работы, цели, оборудование, время выполнения работы, порядок выполнения работы, вывод по работе.

Порядок выполнения работы должен включать краткое описание произведенных действий, с использованием (при необходимости) иллюстративных материалов.

## **4 Учебно-методическое и информационное обеспечение**

### **Основные источники:**

1. Казанский, А. А. Прикладное программирование на Excel 2019 : учебное пособие для среднего профессионального образования / А. А. Казанский. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2022. — 171 с. — (Профессиональное образование). — ISBN 978-5-534-12461-3. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/490886>
2. Казанский, А. А. Программирование на Visual C# : учебное пособие для среднего профессионального образования / А. А. Казанский. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2022. — 192 с. — (Профессиональное образование). — ISBN 978-5-534-14130-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/491341>
3. Подбельский, В. В. Программирование. Базовый курс C# : учебник для среднего профессионального образования / В. В. Подбельский. — Москва : Издательство Юрайт, 2022. — 369 с. — (Профессиональное образование). — ISBN 978-5-534-11467-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/475775>

### **Дополнительные источники:**

1. Методические рекомендации по выполнению самостоятельной работы по дисциплине «Системное программирование» для студентов специальности 09.02.03 Программирование в компьютерных системах, 2022г.
2. Методическое учебное пособие по дисциплине «Системное программирование» для студентов специальности 09.02.03 Программирование в компьютерных системах, 2022г.
3. Методическое учебное пособие по дисциплине «Прикладное программирование» для студентов специальности 09.02.03 Программирование в компьютерных системах, 2022г.
4. Методические рекомендации по выполнению самостоятельной работы по дисциплине «Прикладное программирование» для студентов специальности 09.02.03 Программирование в компьютерных системах, 2022г.
5. Нагаева, И. А. Программирование: Delphi : учебное пособие для вузов / И. А. Нагаева, И. А. Кузнецов ; под редакцией И. А. Нагаевой. — Москва : Издательство Юрайт, 2022. — 302 с. — (Высшее образование). — ISBN 978-5-534-07098-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/493669>

6. Лачуга, Ю. Ф. Прикладная математика : учебник и практикум для среднего профессионального образования / Ю. Ф. Лачуга, В. А. Самсонов. — 2-е изд., доп. — Москва : Издательство Юрайт, 2022. — 304 с. — (Профессиональное образование). — ISBN 978-5-534-13214-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/494915>
7. Казанский, А. А. Объектно-ориентированный анализ и программирование на Visual Basic 2013 : учебник для среднего профессионального образования / А. А. Казанский. — Москва : Издательство Юрайт, 2022. — 290 с. — (Профессиональное образование). — ISBN 978-5-534-03833-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/491340>