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»

Ставрополь 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Целью изучения дисциплины «Современные технологии программирования» является формирование навыков по разработке, документиро-ванию и сопровождению сетевых приложений, расширение профессиональ-ного кругозора студентов, повышение программистской культуры. формиро-вание 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 функциони-рования Internet-приложений и навыков их разработки,

2) изучение основных подходов, платформ, технологий и инструментов проектирования Internet-приложений;

3) формирование набора профессиональных компетенций.

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-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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

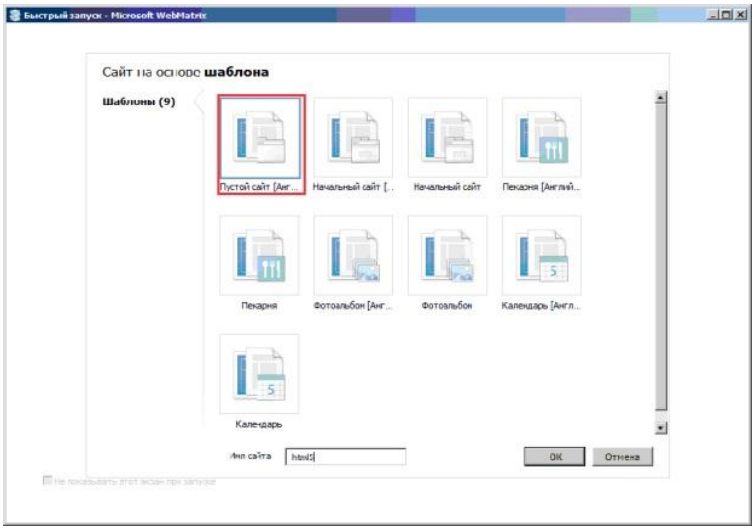


Рисунок 1.2 – Создание сайта на основе пустого шаблона

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

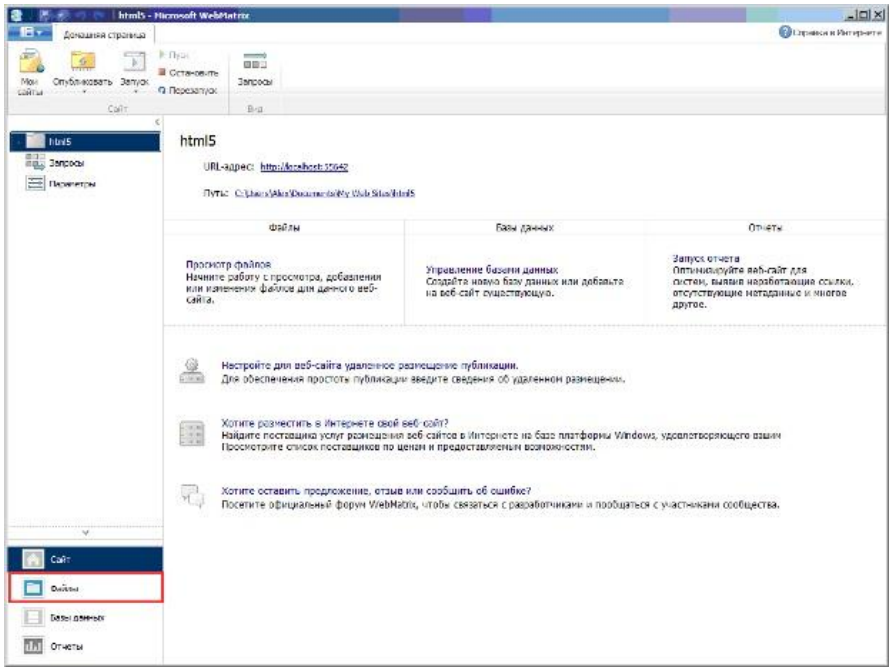


Рисунок 1.3 – Переход к просмотру и изменению файлов сайта

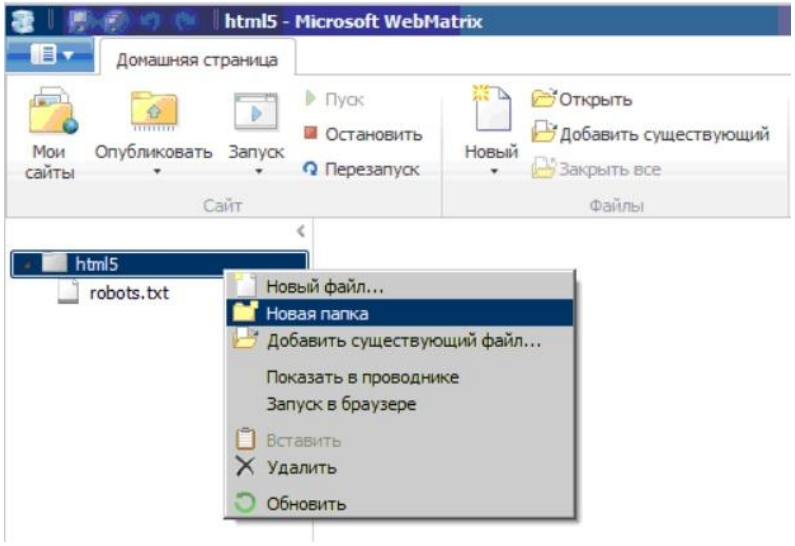


Рисунок 1.4 – Создание новой папки

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).

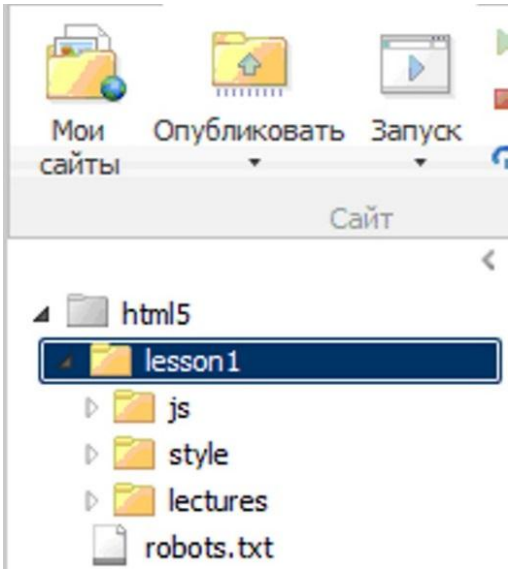


Рисунок 1.5 – Иерархия каталогов

Создание страниц - содержимого

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

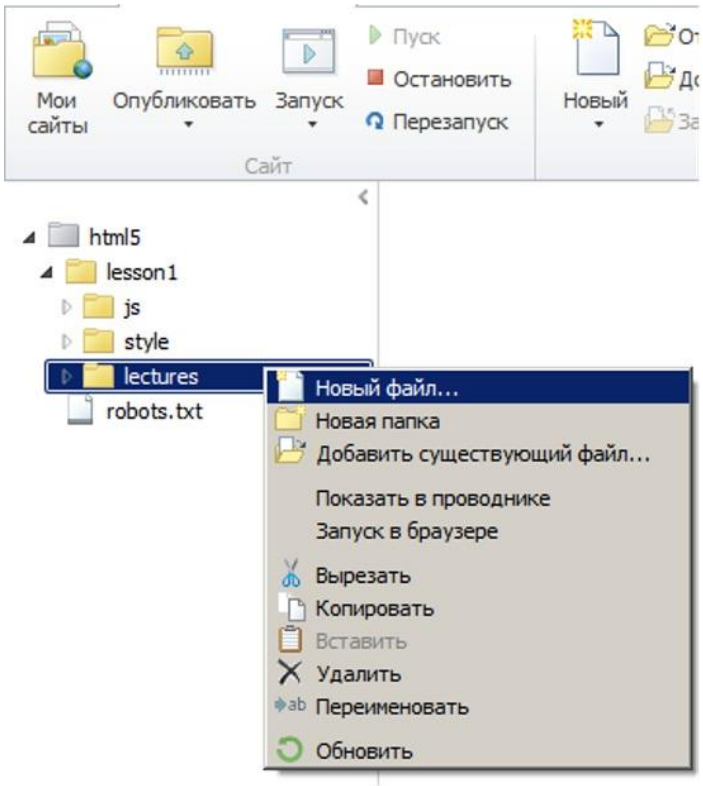


Рисунок 1.6 – Добавление нового файла в папку

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

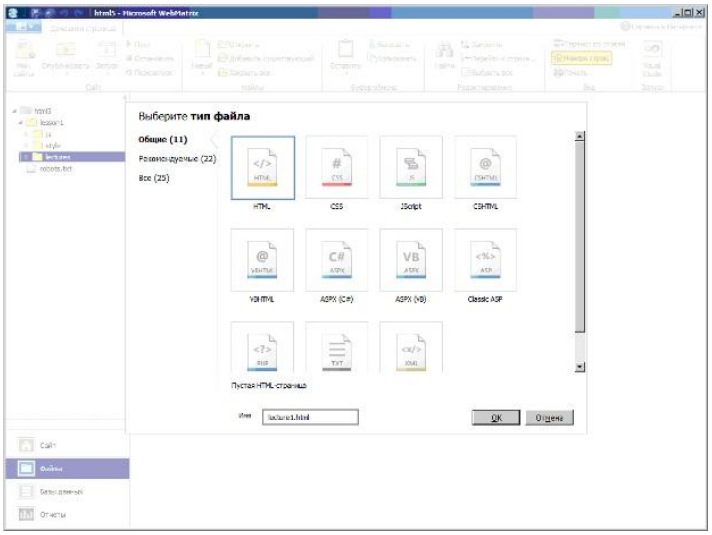


Рисунок 1.7 – Выбор типа создаваемого файла

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.

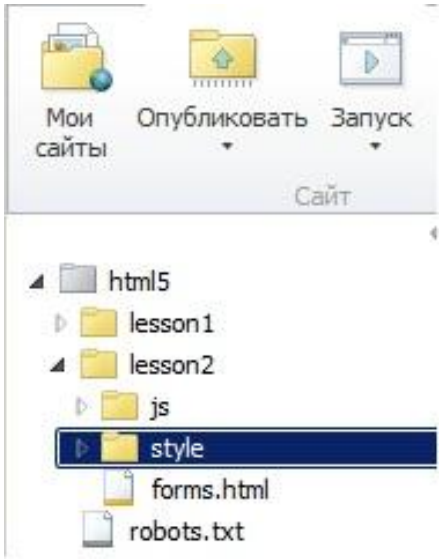


Рисунок – 1.8 – Иерархия каталогов

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:

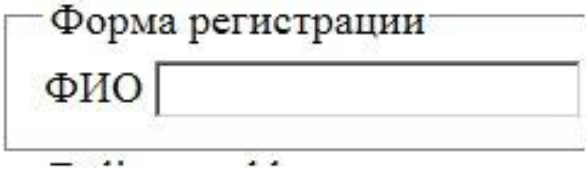


Рисунок 1.9 – Промежуточный результат на шаге 2

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> <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:

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:

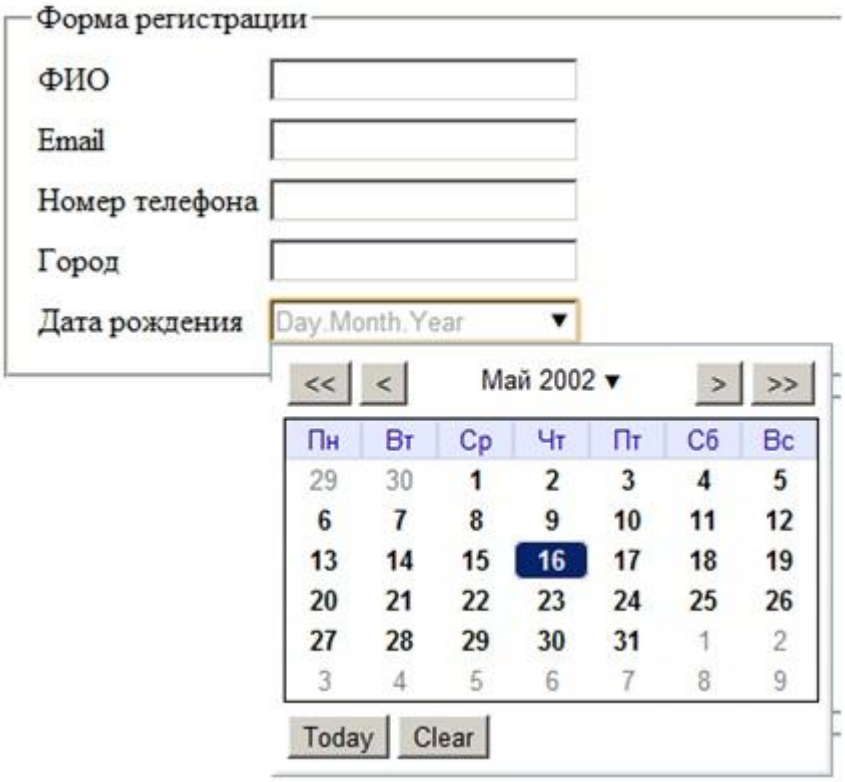


Рисунок 1.12 – Промежуточный результат на шаге 5

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.

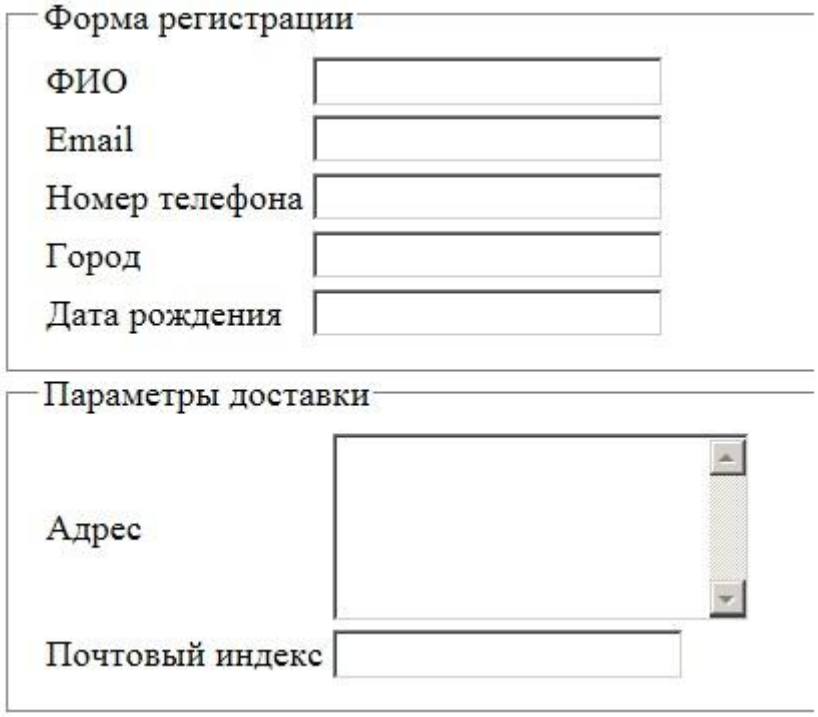


Рисунок 1.13 – Промежуточный результат на шаге 6

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>
```

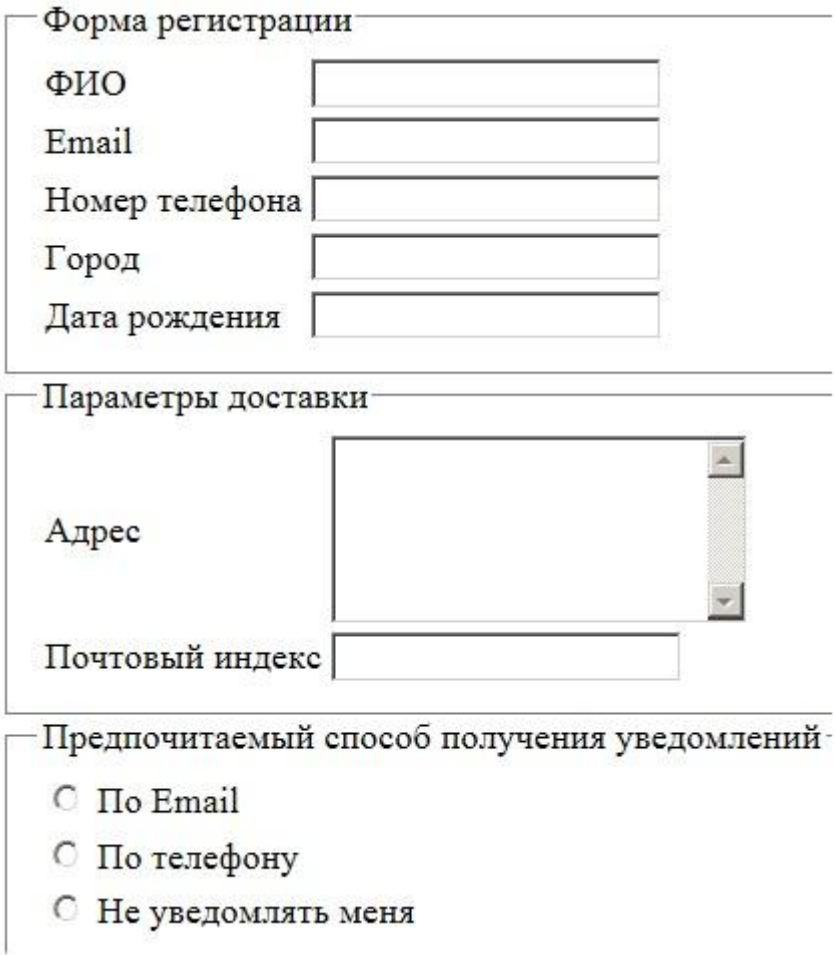


Рисунок 1.14 – Промежуточный результат на шаге 7

Шаг 8. Добавление на страницу кнопки

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

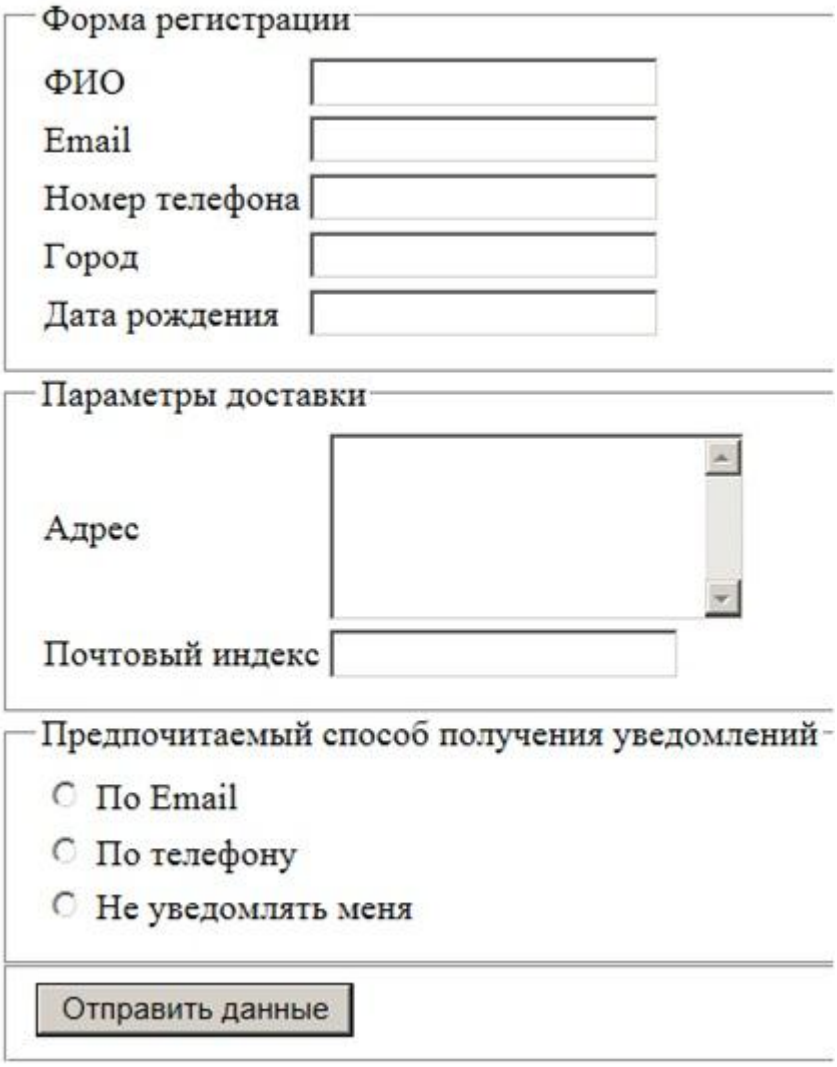


Рисунок 1.15 – Промежуточный результат на шаге 8

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. Выровняем 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>
 <tr>
  <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 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>
```

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. Выровняем 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 сопадает с общепринятой для работы с персональным компютером. 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 образу многочисленных регистрационных Web-страниц, вида представленного на рисунке.

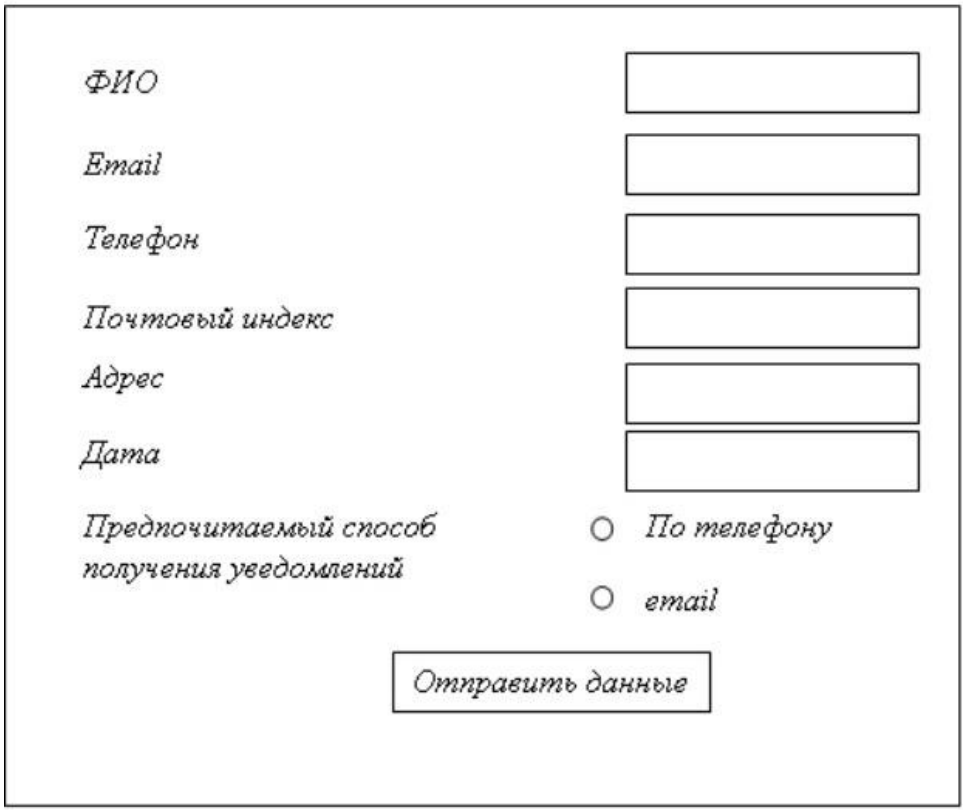
Необходимо также создать соответствующие стили для оформления Web-формы и валидацию вводимых значений.

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».



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?

СПИСОК ЛИТЕРАТУРЫ: основная – 2, 3, 4, 5; дополнительная – 2, 3.

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:

1. ОПК-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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="conteiner" >
<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="conteiner"></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;
}
  .conteiner {
```

```
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 элементом 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 ние: 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="conteiner" ondragenter="return dragEnter(event)"
ondrop="return dragDrop(event)" ondragover="return dragOver(event)">
    <div id="firstDragElement" class="element" ondragstart="return dragStart(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 dragStart(event)"
ondragend="return dragEnd(event)" draggable="true">Text 3</div>
</div>
<div id="target" class="conteiner" ondragenter="return dragEnter(event)"
ondrop="return dragDrop(event)" ondragover="return dragOver(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>
  <img id=”poster” src=”audio\Ray Charles\The Genius Hits The Road\AlbumArtSmall.jpg”/>
  <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</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</td>

<td><a class=”listenlink” onclick=”testListen

(‘audio/Ray Charles/The Genius Hits The Road/03-Basin Street Blues.mp3’);

“>Listen</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</td>

<td><a class=”listenlink” onclick=”testListen

(‘audio/Ray Charles/The Genius Hits The Road/04-Mississippi Mud.mp3’);

“>Listen</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 пользовательскую 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.addtoPLlink: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 добавления трека в список, необходимо явно перезагрузить основную страницу*/

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 element*/

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 будет отображаться*/

div.appendChild(audio); /*добавляем созданный audio элемент контейнеру*/

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 находм 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');
 /*инициируем 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 сопадает с общепринятой для работы с персональным компютером. 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 контейне-рами;

- 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.

СПИСОК ЛИТЕРАТУРЫ: основная – 2, 3, 4, 5; дополнительная – 2, 3.

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:

1. ОПК-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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 функции 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(){
```

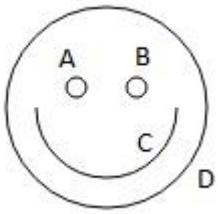
```
var canvas = document.getElementById('text');
var ctx = canvas.getContext('2d');
ctx.beginPath();
ctx.moveTo(20,20); // задаем начальную точку для "пера"
ctx.lineTo(20, 120); // сторона A
ctx.lineTo(120,120); // строна B
ctx.lineTo(20,20); // стророна C
ctx.stroke(); // рисуем незаполненную фигуру
```

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 A
 ctx.lineTo(120,120); // сторона B
 ctx.lineTo(20,20); // сторона C
 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); // строна E
 ctx.lineTo(40,20); // строна 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:

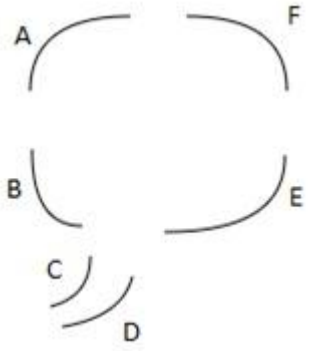


Основная сложность этого задания в переносе "пера" в начальные позиции для рисования, поэтому обратите внимание на координаты, указанные в методах moveTo и arc:

```
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 A к F, то понадобится только один вызов функции moveTo для задания начальных координат для первой кривой (A). 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 A
  ctx.quadraticCurveTo(25,100,50,100); // кривая B
  ctx.quadraticCurveTo(50,120,30,125); // кривая C
  ctx.quadraticCurveTo(60,120,65,100); // кривая D
  ctx.quadraticCurveTo(125,100,125,62.5); // кривая E
  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 сопадает с общепринятой для работы с персональным компютером. 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.

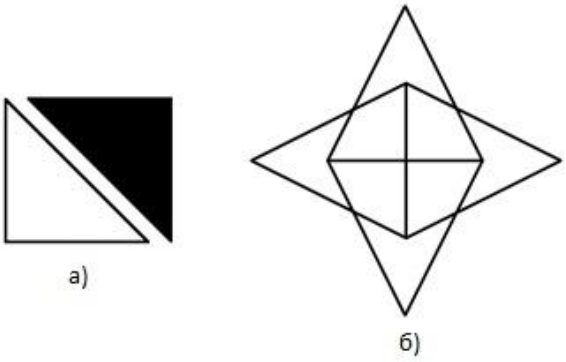
Задания

Задание 1. Разместите элемент <canvas> на странице.

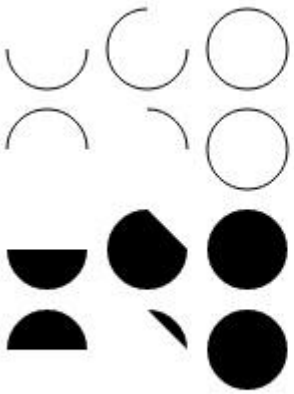
Задание 2. Создать файл стилей для <canvas>.

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

1. Рисование путей



2. Дуги

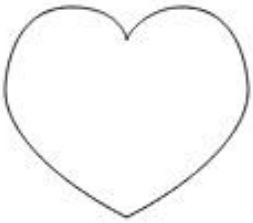


а)

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».

СПИСОК ЛИТЕРАТУРЫ: основная – 2, 3, 4, 5; дополнительная – 2, 3

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:

1. ОПК-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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 – общеязыковой 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; опера-

торы 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 Studia .Net.

Указания по технике безопасности. Техника безопасности при выполнении лабораторной работы сопадает с общепринятой для работы с персональным компютером. 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 сим- волов 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#?

СПИСОК ЛИТЕРАТУРЫ: основная – 1, 2, 4, 6, 7, 8; дополнительная – 1, 2, 4.

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-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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 как коллекции и использовать цикл foreach для перебора его элементов.

Таблица 5.1 – Основные свойства класса System.Array

Свойство	Описание
IsFixedSize	Возвращает true, если массив является статическим

IsReadOnly	Для всех массивов возвращает значение false
IsSynchronized	Возвращает true или false, в зависимости от того, установлена ли синхронизация доступа для массива
Length	Число элементов в массиве
Rank	Размерность массива

Таблица 5.2 – Основные методы класса System.Array

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

Строки

В языке C# определен класс char[], и его можно использовать для представления 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: IComparable, ICloneable, IConvertible, IEnumerable.

Таблица 5.3 – Члены класса

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

Указания по технике безопасности. Техника безопасности при выполнении лабораторной работы сопадает с общепринятой для работы с персональным компютером. 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.Write("Length of row {0} is {1}:\t", i, JagArr[i].Length);
 for (int j = 0; j < JagArr[i].Length; j++)
 {
  Console.Write(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.Write("\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.Write("\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 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.Write("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

СПИСОК ЛИТЕРАТУРЫ: основная – 1, 2, 4, 6, 7, 8; дополнительная – 1, 2, 4

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:

1. ОПК-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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 HttpApplicationState
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 Studia .Net.

Указания по технике безопасности. Техника безопасности при выполнении лабораторной работы сопадает с общепринятой для работы с персональным компютером. 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 значений
    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 и 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.

СПИСОК ЛИТЕРАТУРЫ: основная – 1, 2, 4, 6, 7, 8; дополнительная – 1, 2, 4.

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-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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-эле-мент. Кроме того, все серверные элементы управления HTML должны быть размещены внутри области действия тэга <form>, также имеющего атрибут runat="server".

Таблица 7.1

Серверный элемент управления HTML	**Описание**
HtmlAnchor	Управление HTML элементом <a>
HtmlButton	Управление HTML элементом <button>
HtmlForm	Управление HTML элементом <form>
HtmlGeneric	Управляет HTML элементами не описываемыми как элементы управления HTML , например, <body>, , и др.
HtmlImage	Управление HTML элементом <image>
HtmlInputButton	Управление HTML элементами <input type="button">, <input type="submit"> и <input type="reset">
HtmlInputCheckBox	Управление HTML элементом <input type="checkbox">
HtmlInputFile	Управление HTML элементом <input type="file">
HtmlInputHidden	Управление HTML элементом <input type="hidden">
HtmlInputImage	Управление HTML элементом <input type="image">
HtmlInputRadioButton	Управление HTML элементом <input type="radio">
HtmlInputText	Управление HTML элементами <input type="text"> и <input type="password">
HtmlSelect	Управление HTML элементом <select>

HtmlTable	Управление HTML элементом <table>
HtmlTableCell	Управление HTML элементами <td> и <th>
HtmlTableRow	Управление HTML элементом <tr>
HtmlTextArea	Управление HTML элементом <textarea>

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

Элемент управления для проверки данных	**Описание**
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 Studia .Net.

Указания по технике безопасности. Техника безопасности при выполнении лабораторной работы сопадает с общепринятой для работы с персональным компютером. 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 вода 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).

СПИСОК ЛИТЕРАТУРЫ: основная – 1, 2, 4, 6, 7, 8; дополнительная – 1, 2, 4.

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-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

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 пользователя 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 большинства 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 Studia .Net.

Указания по технике безопасности. Техника безопасности при выполнении лабораторной работы сопадает с общепринятой для работы с персональным компютером. 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="ASPNETGView2._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" AutoGenerateCol-
umns="False"
        BackColor="White" BorderColor="#E7E7FF" BorderStyle="None" Bor-
derWidth="1px"
        CellPadding="3" DataKeyNames="id" Data-
SourceID="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" SortExpres-
sion="Title" />
          <asp:BoundField DataField="Teacher" HeaderText="Teacher"
            SortExpression="Teacher" />
          <asp:BoundField DataField="Type" HeaderText="Type" SortExpres-
sion="Type" />
        </Columns>
```

```
        <PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" Horizonta-
lAlign="Right" />
        <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeCol-
or="#F7F7F7" />
        <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeCol-
or="#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" CellPadding="3"
    CellSpacing="1" GridLines="None" >
    <FooterStyle BackColor="#C6C3C6" ForeColor="Black" />
    <SelectedItemStyle BackColor="#9471DE" Font-Bold="True" ForeColor="White" />
    <PagerStyle BackColor="#C6C3C6" ForeColor="Black" HorizontalAlign="Right" />
    <ItemStyle BackColor="#DEDFDE" ForeColor="Black" />
    <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#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="ASPNETGView2._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 DataL-ist.

СПИСОК ЛИТЕРАТУРЫ: основная – 1, 2, 4, 6, 7, 8; дополнительная – 1, 2, 4.

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 образова-

ния 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. Храмцов, 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 ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению самостоятельной работы
по дисциплине «Технологии разработки Internet-приложений»
для направления подготовки 09.04.02 «Информационные системы и технологии»
(магистерская программа «Управление данными»)

Ставрополь 2017

Введение

Целью изучения дисциплины «Технологии разработки Internet-приложений» 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 09.04.02 «Информационные системы и технологии».

Задачами изучения дисциплины «Технологии разработки Internet-приложений» 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.

Дисциплина относится к блоку Б1 вариативная часть, дисциплины по выбору студента – Б1.В.ДВ.3.2. Ее освоение происходит в 3 семестре.

При изучении дисциплины «Технологии разработки Internet-приложений» могут использоваться знания, полученные в ходе освоения курсов «Математическое обеспечение систем поддержки принятия решений», «Современные информационные технологии в науке и образовании».

Знания, полученные в ходе изучения курса «Технологии разработки Internet-приложений», могут быть использованы при подготовке отчетов по научно-исследовательской работе, государственной итоговой аттестации.

В ходе изучения дисциплины «Технологии разработки Internet-приложений» у обучающегося формируются следующие компетенции:

1. ОПК-5 – владением методами и средствами получения, хранения, переработки и трансляции информации посредством современных компьютерных технологий, в том числе, в глобальных компьютерных сетях.

2. ПК-3 – умением разрабатывать новые технологии проектирования информационных систем.

3. ПК-15 – способностью разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач.

В соответствии с целями, задачи дисциплины и компетенциями формируемой ею, в результате ее освоения дисциплины обучающийся должен

Знать:

- основные определения: программа, приложение; Internet-приложение; обогащенное Internet-приложение; система управления контентом;

- классификацию Web-сайтов, Web-серверов и Internet- приложений (Web-приложений);

- принципы работы Internet-приложений; задачи, решаемые с помощью Internet-приложений; средства и технологии проектирования Internet-приложений;

- принципы разработки методов решения нестандартных и традиционных задач;

- принципы формирования новые конкурентоспособных идей в области теории и практики информационных технологий и систем;

- новые методы и средства проектирования информационных систем.

Уметь:

- сформулировать определения основных понятий, использующихся при изучении курса «Технологии разработки Internet-приложений»;

- дать характеристику основным видам Internet-приложений;

- разрабатывать новые методы, средства и технологии проектирования информационных систем;

- разрабатывать методы решения нестандартных задач и новые методы решения традиционных задач;

- формировать новые конкурентоспособные идеи в области теории и практики информационных технологий и систем.

Владеть:

- инструментальными средствами для разработки Internet-приложений;

- основными технологиями разработки Internet-приложений;

- инструментарием сетевых технологий;

- методами решения нестандартных задач и новые методами решения традиционных задач;

- умением разрабатывать новые методы и средства и технологии проектирования информационных систем

Содержание дисциплины

Раздел 1. Введение в технологию разработки Internet-приложений

Тема 1. Internet-приложения. Основные определения и понятия.

Тема 2. Internet-приложения. Классификация Web-приложений.

Тема 3. Обзор Web-серверов.

Раздел 2. Технологии разработки Internet-приложений

Тема 4. Клиентские сценарии и приложения

Тема 5. Серверные Web-приложения.

Тема 6. Общие сведения о технологиях ASP

Тема 7. Введение в технологию AJAX.

Тема 8. XML как технология разработки Internet-приложений

1. Организационно-методические рекомендации по освоению дисциплины

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого из лабораторных занятий и включает изучение теоретического материла, приведенного в ней, а также выполнения индивидуальных заданий, приведенных в лабораторной работе.

Подготовленный материал, практическое задание оформляются в конспекте.

Самостоятельные занятия (СЗ) являются одной из активных форм обу-

чения.

Самостоятельные занятия по дисциплине «Технологии разработки Internet-приложений» имеют целью:

- закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных занятий;

- сформировать практические навыки работы с инструментальными средствами для решения задач приближенными (численными) методами; использования математических методов обработки, анализа и синтеза результатов профессиональных исследований; методами численного моделирования процессов и систем.

Самостоятельные занятия проводятся в лабораториях, оборудованных современными средствами вычислительной техники и возможностью выхода в сеть Internet, в библиотеке.

Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и защите лабораторных работ по дисциплине «Технологии разработки Internet-приложений».

2. Методические указания по выполнению самостоятельной работы студентов

Методические указания должны включать следующие разделы:

- цель работы;

- задание, которое должно быть выполнено студентом в результате проведения самостоятельной работы;

- варианты индивидуальных заданий;

- основные теоретические положения, необходимые для выполнения задания, они должны быть краткими и содержать ссылки на литературу, в которой эти положения изложены в объеме, достаточном для выполнения самостоятельной работы;

- этапы выполнения задания с указанием конкретных сроков выполне-

ния каждого из этапов и всего задания в целом;

- требования к оформлению графической и текстовой части самостоятельной работы;

- пример выполнения одного из вариантов задания и оформления отчета;

- библиографический список использованных источников.

3. Содержание самостоятельной работы

3.1 Самостоятельное изучение тем

Раздел 1. Введение в технологию разработки Internet-приложений

Тема 1: «*Обзор Web-серверов»*

1. Общее представление о Web-сервере.
2. Сервер Apache.
3. Сервер Microsoft Internet Information Server. Характеристика.

Тема 2: «Web-порталы»

1. Портал. Портлеты. Веб-портал.
2. Классификация порталов: горизонтальные порталы, вертикальные порталы, корпоративные порталы.

Тема 3: «HTML5 Accessibility»

1. WCAG
2. WAI-ARIA. Применение WAI-ARIA
3. ARIA-роли

Тема 4: «SVG – векторная графика в WWW»

1. Язык анимации SVG – SMIL
2. Библиотеки для работы с SVG
3. Canvas vs SVG

Тема 10: «*Приложения для социальных сетей»*

1. Понятие «Социальный Web».

2. Социальное программное обеспечение.

3. Программные системы, относящиеся к социальному программному обеспечению.

Тема 11: *«Введение в Web 2.0»*

1. Web 2.0: ключевые технологии и недостатки.

2. Мэшапы: архитектура и классификация.

Раздел 2. Технологии разработки Internet-приложений

Тема 5: «Храним данные на клиенте WebStorage / WebSQL / WebNoSQL»

1. Хранилище «ключ/значение» в браузере WebStorage

2. Реляционная база данных на Web-странице WebSQL

Тема 6: «Клиентские сценарии и приложения»

1. Программы, выполняющиеся на клиент-машине.

2. Программы, выполняющиеся на сервере.

3. Введение в Jscript: типы данных, операторы, функции и объекты.

4. Краткая характеристика VBScript.

5. Java-апплеты.

6. ActionScript-общая характеристика.

Тема 7: «Серверные Web-приложения»

1. Стандарт CGI.

2.Сценарии.

3. Сценарные языки: классфикация по быстродействию.

Тема 8: «Организация процесса разработки Web-контента. CMS/CMF»

1. Система управления контентом (CMS).
2. Система управления Web-контентом (WCMS).
3. Типы WCMS-систем.

Повышенный уровень

Раздел 1. Введение в технологию разработки Internet-приложений

Тема 1: «Обзор Web-серверов»

1. Использование Web-серверов: Microsoft Internet Information Server; Personal Web Server.

Тема 3: «HTML5 Accessibility»

1. ARIA-роли

Тема 4: «SVG – векторная графика в WWW»

1. Canvas vs SVG

Тема 10: «Приложения для социальных сетей:

1. Фолксономия.
2. Семантическая Web-сеть.
3. Онтология.
4. Семантические Web-сервисы.

Тема 11: «Введение в Web 2.0»

1. Мэшапы: архитектура и классификация.

Раздел 2. Технологии разработки Internet-приложений

Тема 5: «Храним данные на клиенте WebStorage / WebSQL / WebNoSQL»

1. NoSQL в Web – IndexedBD

Тема 6: «Клиентские сценарии и приложения»

1. XAML и Microsoft Silverlight.
2. Понятие о DOM.
3. DHTML.

Тема 7: «Серверные Web-приложения»

1. Язык Python.
2. Язык Ruby.
3. Интерфейс ISAPI.

Тема 7: «Организация процесса разработки Web-контента. CMS/CMF»

1. WCMS Drupal.

Тема 9: «*Синдикация и аггрегирование Web-контента»*

1. Web-синдикация.
2. Web-поток. Аггрегатор потоков. Преимущества Web-потоков.
3. RSS.

В процессе самостоятельного изучения тем лекционного курса студент изучает предложенные их по рекомендуемому преподавателем списку литературы и предоставляет преподавателю краткий конспект, по которому в дальнейшем проводится индивидуальное собеседование.

3.2 Коллоквиум

Коллоквиум – это форма проверки и оценивания знаний учащихся в системе образования и представляет собой мини-экзамен, проводимый в середине семестра и имеющий целью уменьшить список тем, выносимых на дифференцированный зачет или экзамен.

Коллоквиум как одна из форм самостоятельной работы решает следующие задачи:

- проверка и контроль знаний, полученных студентами по изучаемой теме;

- расширение проблематики в рамках дополнительных вопросов по данной теме;

- углубление знаний при помощи дополнительных материалов при подготовке к занятию;

- студенты должны продемонстрировать умения работать с различными видами литературных источников;

- формирование навыков коллективного обсуждения (поддерживать диалог в микрогруппах, находить компромиссное решение и т.д.).

Коллоквиум может проводиться в двух формах – устной и письменной.

При проведении коллоквиума в устной форме ответы студентов оцениваются по традиционной шкале («неудовлетворительно» – «отлично») по билетам, которые могут содержать как теоретические вопросы, так и практические задания. На коллоквиум рекомендуется выносить лишь часть теоретического материала. Полученная по итогам его сдачи оценка влияет на итоговую по дисциплине.

Коллоквиум в письменной форме также проводится по билетам, в который могут быть включены теоретические вопросы, предполагающие короткий ответ, а также практические задания.

Критериями оценки знаний студентов на коллоквиуме являются:

Оценка «отлично» может быть выставлена, если студент демонстрирует глубокое и прочное усвоение программного материала, дает полные, по-

следовательные грамотные и логически излагаемые ответы при видоизменении задания, свободно справляется с поставленными задачами и правильно применяет знания материала, четко обосновывает принятые решения, владеет разносторонними навыками и приемами выполнения практических работ.

Оценка «хорошо» по итогам сдачи коллоквиума студент может получить, если показывает хорошее знание программного материала, грамотно излагает без существенных неточностей в ответ на поставленный вопрос, правильно применяет теоретические знания, владеет необходимыми навыками при выполнении практических задач.

Оценка «удовлетворительно» может быть выставлена, если студент показал удовлетворительный уровень усвоения основного материла, допускает неточности при ответе на поставленные вопросы и в ответе присутствуют недостаточно правильные формулировки, нарушает последовательность в изложении программного материала, испытывает затруднения в выполнении практических заданий

Оценка «неудовлетворительно» по итогам сдачи коллоквиума выставляется, если студент демонстрирует полное незнание программного материала, допускает при ответе на вопрос и выполнении практических заданий грубые ошибки.

В процессе изучения дисциплины «Технологии разработки Internet-приложений» студент должен сдать коллоквиум по разделу: «*Технологии разработки Internet-приложений».*

Коллоквиумы по дисциплине «Технологии разработки Internet-приложений» проводится в форме собеседования в устной форме по билетам, в который включены два вопроса, один из которых является вопросом базового, а второй повышенного уровня.

Вопросы к коллоквиуму

Раздел 2 «Технологии разработки Internet-приложений»

Базовый уровень

1. Клиентские сценарии и приложения. Программы, выполняющиеся на клиент-машине.

2. Клиентские сценарии и приложения. Программы, выполняющиеся на сервере.

3. Введение в Jscript: типы данных.

4. Введение в Jscript: операторы.

5. Введение в Jscript: функции и объекты.

6. Краткая характеристика VBScript.

7. Java-апплеты.

8. Стандарт CGI.

9. Сценарии.

10. Сценарные языки: классфикация по быстродействию.

11. Интерфейс ISAPI.

12. Принципы функционирования Active Server Page.

13. История технологий ASP.

14. Общие сведения о технологиях ASP.NET.

15. Процесс создания Web-приложения на ASP.NET.

16. Структура и история развития технологии AJAX.

17..Объект XmlHttpRequest.

18. Инструментарий разработки AJAX-приложений

19. Ведение в технологию XML.

20. Структура XML-документа.

21. Основные синтаксические правила построения XML-документов

Повышенный уровень

1 ActionScript-общая характеристика.

2. XAML и Microsoft Silverlight.

3. Понятие о DOM.

4. DHTML.

5. Язык Python.

6. Язык Ruby.

7. Модель Active Server Page.

8. Использование дополнительных средств при разработке приложений на основе технологий ASP.NET.

9. Серверные элементы управления ASP.NET.

10. Работа с источниками данных в ASP.Net.

11. Безопасность AIAX-приложений

12. Разработк мобильных Web-приложений

13. Стандартизация основных напрвлений XML-технологий

3.3 Доклад

Изложенное понимание содержания доклада как целостного авторского текста определяет критерии его оценки: новизна текста; обоснованность выбора источника или источников; степень раскрытия предложенной темы; соблюдения требований к оформлению.

Новизна текста:

а) актуальность темы исследования;

б) новизна и самостоятельность в постановке проблемы, формулирование нового аспекта известной проблемы в установлении новых связей (межпредметных, внутрипредметных, интеграционных);

в) умение работать с исследованиями, критической литературой, систематизировать и структурировать материал;

г) явленность авторской позиции, самостоятельность оценок и суждений;

д) стилевое единство текста, единство жанровых черт.

Степень раскрытия сущности вопроса:

а) соответствие плана теме реферата;

б) соответствие содержания теме и плану реферата;

в) полнота и глубина знаний по теме;

г) обоснованность способов и методов работы с материалом;

д) умение обобщать, делать выводы, сопоставлять различные точки

зрения по одному вопросу (проблеме).

Обоснованность выбора источников:

а) оценка использованной литературы: привлечены ли наиболее известные работы по теме исследования.

Соблюдение требований к оформлению:

а) насколько верно оформлены ссылки на используемую литературу, список литературы;

б) оценка грамотности и культуры изложения, владение терминологией;

в) соблюдение требований к объёму доклада.

Оценка «отлично» ставится, если выполнены все требования к написанию и защите доклада: обозначена проблема и обоснована её актуальность, сделан краткий анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объём, соблюдены требования к внешнему оформлению, даны правильные ответы на дополнительные вопросы.

Оценка «хорошо» может быть выставлена, если основные требования к докладу и его защите выполнены, но при этом допущены недочёты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объём доклада; имеются упущения в оформлении; на дополнительные вопросы при защите даны неполные ответы.

Оценка «удовлетворительно» может быть выставлена, если имеются существенные отступления от требований к реферированию. В частности, тема освещена лишь частично; допущены фактические ошибки в содержании доклада или при ответе на дополнительные вопросы; во время защиты отсутствует вывод.

Оценка «неудовлетворительно» может быть выставлена, если тема доклада не раскрыта, обнаруживается существенное непонимание проблемы.

Вся тематика докладов по дисциплине «Технологии разработки

Internet-приложений» разбита на две группы – базового и повышенного уровня, таким образом, главное различие этих заданий, во-первых, заключается в самой тематики, а, во-вторых, подходом к изложению темы. В пояснительных записках к докладам на темы повышенного уровня необходимо помимо теории так же привести практические примеры.

Процедура проведения данного оценочного мероприятия включает в себя собеседование по подготовленному студентами докладу.

В процессе подготовки доклада по дисциплине необходимо изучить список предлагаемой преподавателем литературы, самостоятельно разработать план доклада, осуществить поиск информации с помощью глобальной сети Internet.

4. План-график выполнения СРС

№№	Форма самостоятельной работы	Срок ее сдачи
1	Индивиуальное собеседование по результатам самостоятельного изучения тем лекций	по итогам изучения соответствующего раздела лекционного курса
2	Коллоквиум	по итогам изучения раздела курса «*Технологии разработки Internet-приложений».*
3	Доклад	по итогам изучения лекционного курса

5. Организация контроля знаний студентов

5.1. Формы контроля знаний студентов

Контроль и оценка знаний, умений и навыков студентов осуществляется на лабораторных занятиях, консультациях. В ходе контроля знаний преподаватель оценивает понимание студентом содержания дисциплины «Технологии разработки Internet-приложений», его способность разрабатывать и использовать новые методы, средства и технологии для проектирования, разработки и поддержания функционирования

Контроль знаний студентов может осуществляться в следующих формах:

- текущий контроль знаний;
- итоговый контроль знаний.

Текущий контроль знаний студентов имеет целью:

- дать оценку работы каждого студента по усвоению им учебного материала, выявить недостатки в его подготовке и оказать практическую помощь в их устранении;

Основными формами текущего контроля знаний студентов являются:

- устный контрольный опрос;
- письменный контрольный блиц-опрос;
- защита лабораторной работы;
- проверка конспектов лекций.

Устный контрольный опрос студентов проводится на лекциях (и лабораторных занятиях). По его результатам преподаватель оценивает качество подготовки студента к занятию.

Письменный контрольный блиц-опрос студентов проводится в течение пяти минут на лабораторных занятиях путем письменного ответа их на пять вопросов, заданных преподавателем. Результаты его проведения отмечаются в журнале. На лабораторных занятиях знания и практические навыки студен-

тов оцениваются по 5-балльной системе. Полученные оценки выставляется в журнале.

При проверке конспектов лекций дается анализ качества их ведения. Отмечаются допущенные ошибки, в рецензии преподавателя оценивается качество конспектирования учебного материала, даются рекомендации по улучшению качества конспектирования лекционного материала.

5.2. Рекомендации по подготовке к зачету

Подготовка к зачету начинается с начала изучения дисциплины. Для получения зачета необходимо посещать все лекции и лабораторные занятия, а так же предоставить отчеты о выполнении лабораторных работ, сдать коллоквиум, выполнить доклад и индивидуальное домашнее задание, пройти их защиту.

6. Рекомендации по работе с литературой и источниками

6.1. Рекомендации студентам по организации работы с литературой и источниками

Изучение литературы и источников необходимо начинать с прочтения соответствующих глав учебных изданий, учебных пособий или литературы, рекомендованной в качестве основной или дополнительной по дисциплине «Технологии разработки Internet-приложений», которые прямо или косвенно относятся к изучаемой теме.

Изучая литературу и источники, студенту рекомендуется вести краткий конспект. Однако не следует переписывать все содержание изучаемой темы, нужно выписывать лишь основные идеи и главные на ваш взгляд мысли. В отдельных случаях, когда встречаются важные определения, понятия, необходимый фактический материал и примеры, статистическая информация, имеющие отношение к изучаемой теме, студенту следует выписать их в виде

цитат с полным указанием библиографических источников.

Конспектирование рекомендуемой литературы и источников необходимо вести с распределением собранных материалов по отдельным главам и параграфам согласно учебно-тематическому плану. Необходимо выписывать все выходные данные по используемой литературе и источникам.

Основой технологии интенсификации обучения на платформе цифровых образовательных технологий являются учебно-иллюстрационные материалы (опорный конспект) по дисциплине «Технологии разработки Internet-приложений».

Работа с учебно-иллюстрационными материалами имеет следующие этапы.

1. Изучение теоретических основ учебного материала в аудитории: изложение преподавателем изучаемого материала студентам с объяснением по опорному конспекту;

2. Самостоятельная работа: индивидуальная работа студентов по опорному конспекту; фронтальное закрепление по блокам опорного конспекта.

3. Первое повторение – воспроизведение содержания заданной темы опорного конспекта по памяти.

4. Устное проговаривание материала опорного конспекта – необходимый этап внешнеречевой деятельности при усвоении учебного материала.

5. Второе повторение – взаимоопрос и взаимопомощь студентов друг другу.

Применение учебно-иллюстрационных материалов позволяет обобщить сложный по содержанию материал, активизировать мыслительную деятельность студентов.

Необходимо помнить, что главное для студента в самостоятельной работе с рекомендуемой литературой и источниками – это формирование своего индивидуального стиля, который может стать основой в будущей профессиональной деятельности.

6.2. Перечень рекомендуемой литературы

Основная литература

1. Крахоткина, Е. В. Технологии разработки Internet-приложений: учебно-методическое пособие: Направление подготовки. 09.04.02 Информационные системы и технологии. Магистерская программа "Управление данными". Квалификация (степень) выпускника - магистр. / Крахоткина Е. В.; Сев.-Кав. федер. ун-т. - Ставрополь : СКФУ, 2015. - 123 с.

2. 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

3. 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. Сычев, А. В. Перспективные технологии и языки веб-разработки / А. В. Сычев. – 2-е изд., испр. – М.: Национальный Открытый Университет «ИНТУИТ», 2016. – 494 с. [Электронный ресурс]. Режим доступа: http://biblioclub.ru/index.php?page=book&id=429078

2. Богданов, М. Р. Перспективные языки веб-разработки / М. Р. Богданов. – 2-е изд., испр. – М.: Национальный Открытый Университет «ИНТУИТ», 2016. – 265 с. [Электронный ресурс]. Режим доступа: http://biblioclub.ru/index.php?page=book&id=428953

3. 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

4. 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

5. 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

6. 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

7. 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)

8. 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)

Методическая литература

1. Крахоткина Е. В. Технологии разработки Internet-приложений: учебное пособие (лабораторный практикум). – Ставрополь: Изд-во СКФУ, 2016. – 116 с. (передано в Издательско-полиграфический комплекс СКФУ).

2. Крахоткина Е. В. Технологии разработки Internet-приложений**:** учебное пособие. – Ставрополь: Изд-во СКФУ, 2016. – 141 с. (передано в Издательско-полиграфический комплекс СКФУ).

6.3. Internet-ресурсы:

1) http://www.optim.ru/cs/ – Журнал «Технология Клиент-Сервер»

2) http://www.aspnetmania.com – ASP.NET Mania - все про ASP.NET и .NET Framework

3) http://javascripts.boom.ru – JavaScript без границ

4) http://www.dotsite.ru – dotSITE Portal - Все о .NET, C#, VB.NET, ASP.NET, XML…

5) http://www.intuit.ru/department/internet/aspnetsetup/ – Конфигурирование и настройка Microsoft ASP.NET

6) http://www.intuit.ru/department/se/aspdotnet/ – Основы ASP.NET 2.0.

7) http://www.intuit.ru/department/internet/wapwml/ – Основы WAP/WML и WMLScript

8) http://www.intuit.ru/department/internet/xml/ – Основы XML.

9) http://www.intuit.ru/department/internet/jsbasics/ – Основы программирования на JavaScript.

6.4 Материально-техническое обеспечение дисциплины

К материально-техническому уровню освоения содержания дисциплины «Технологии разработки Internet-приложений» можно отнести:

- индивидуальное взаимодействие со студентами в процессе проведения лекционных и лабораторных занятий;

- использование на лекциях и лабораторных занятиях мультимедийного оборудования;

- включение в лабораторные работы индивидуального поиска, систематизации и анализа информации через Internet;

- авторские презентации к лекциям;

- лекционная аудитория должна быть оборудована мультимедиа проектором;

- Microsoft Office - Word;

- программа для создания Web-страниц, например – блокнот;

- программа для просмотра Web-страниц – программы браузеры Opera, Internet Explorer; Google Chrome.

- среда разработки приложений Microsoft Visual Studio.NET;

- ОС Windows;

- компьютерный класс с предустановленным программным обеспечением.

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

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Методические указания к самостоятельным работам

Направление подготовки 09.03.02 Информационные системы и технологии

Направленность (профиль) «Информационные системы и технологии в бизнесе»

Квалификация выпускника – бакалавр

Невинномысск 2022

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 студента 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 Подготовка к лекциям

Главное в период подготовки к лекционным занятиям – научиться методам самостоятельного умственного труда, сознательно развивать свои творческие способности и овладевать навыками творческой работы. Для этого необходимо строго соблюдать дисциплину учебы и поведения. Четкое планирование своего рабочего времени и отдыха является необходимым условием для успешной самостоятельной работы. В основу его нужно положить рабочие программы изучаемых в семестре дисциплин.

Каждому студенту следует составлять еженедельный и семестровый планы работы, а также план на каждый рабочий день. С вечера всегда надо распределять работу на завтрашний день. В конце каждого дня целесообразно подводить итог работы: тщательно проверить, все ли выполнено по намеченному плану, не было ли каких-либо отступлений, а если были, по какой причине это произошло. Нужно осуществлять самоконтроль, который является необходимым условием успешной учебы. Если что-то осталось невыполненным, необходимо изыскать время для завершения этой части работы, не уменьшая объема недельного плана.

Слушание и запись лекций – сложный вид вузовской аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность студента. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект является полезным тогда, когда записано самое существенное, основное и сделано это самим студентом. Не надо стремиться записать дословно всю лекцию. Такое «конспектирование» приносит больше вреда, чем пользы. Запись лекций рекомендуется вести по возможности собственными формулировками. Желательно запись осуществлять на одной странице, а следующую оставлять для проработки учебного материала самостоятельно в домашних условиях.

Конспект лекций лучше подразделять на пункты, параграфы, соблюдая красную строку. Этому в большой степени будут способствовать пункты плана лекции, предложенные преподавателям. Принципиальные места, опре-

деления, формулы и другое следует сопровождать замечаниями «важно», «особо важно», «хорошо запомнить» и т.п. Можно делать это и с помощью разноцветных маркеров или ручек. Лучше если они будут собственными, чтобы не приходилось присить их у однокурсников и тем самым не отвлекать их во время лекции. Целесообразно разработать собственную «маркографию» (значки, символы), сокращения слов. Не лишним будет и изучение основ стенографии. Работая над конспектом лекций, всегда необходимо использовать не только учебник, но и ту литературу, которую дополнительно рекомендовал лектор. Именно такая серьезная, кропотливая работа с лекционным материалом позволит глубоко овладеть знаниями.

2 Подготовка к лабораторным работам

Подготовку к каждому практическому занятию студент должен начать с ознакомления с методическими указаниями, которые включают содержание работы. Тщательное продумывание и изучение вопросов основывается на проработке текущего материала лекции, а затем изучения обязательной и дополнительной литературы, рекомендованную к данной теме. На основе индивидуальных предпочтений студенту необходимо самостоятельно выбрать тему доклада по проблеме и по возможности подготовить по нему презентацию.

Если программой дисциплины предусмотрено выполнение практического задания, то его необходимо выполнить с учетом предложенной инструкции (устно или письменно). Все новые понятия по изучаемой теме необходимо выучить наизусть и внести в глоссарий, который целесообразно вести с самого начала изучения курса. Результат такой работы должен проявиться в способности студента свободно ответить на теоретические вопросы семинара, его выступлении и участии в коллективном обсуждении вопросов изучаемой темы, правильном выполнении практических заданий и контрольных работ.

В зависимости от содержания и количества отведенного времени на изучение каждой темы практическое занятие может состоять из четырех-пяти частей:

1. Обсуждение теоретических вопросов, определенных программой дисциплины.

2. Доклад и/ или выступление с презентациями по выбранной проблеме.

3. Обсуждение выступлений по теме – дискуссия.

4. Выполнение практического задания с последующим разбором полученных результатов или обсуждение практического задания.

5. Подведение итогов занятия.

Первая часть – обсуждение теоретических вопросов – проводится в виде фронтальной беседы со всей группой и включает выборочную проверку преподавателем теоретических знаний студентов. Примерная продолжительность — до 15 минут. Вторая часть — выступление студентов с докладами, которые должны сопровождаться презентациями с целью усиления наглядности восприятия, по одному из вопросов практического занятия. Обязательный элемент доклада – представление и анализ статистических данных, обоснование социальных последствий любого экономического факта, явления или процесса. Примерная продолжительность — 20-25 минут. После докладов следует их обсуждение – дискуссия. В ходе этого этапа практического занятия могут быть заданы уточняющие вопросы к докладчикам. Примерная продолжительность – до 15-20 минут. Если программой предусмотрено выполнение практического задания в рамках конкретной темы, то преподавателями определяется его содержание и дается время на его выполнение, а замет идет обсуждение результатов. Подведением итогов заканчивается практическое занятие.

В процессе подготовки к практическим занятиям, студентам необходимо обратить особое внимание на самостоятельное изучение рекомендованной учебно-методической (а также научной и популярной) литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у студентов свое отношение к конкретной проблеме. Более глубокому раскрытию вопросов способствует знакомство с дополнительной литературой, рекомендованной преподавателем по каждой теме семинарского или практического занятия, что позволяет студентам проявить свою индивидуальность в рамках выступления на данных занятиях, выявить широкий спектр мнений по изучаемой проблеме.

3 Самостоятельное изучение темы. Конспект

Конспект – наиболее совершенная и наиболее сложная форма записи. Слово «конспект» происходит от латинского «conspectus», что означает «обзор, изложение». В правильно составленном конспекте обычно выделено самое основное в изучаемом тексте, сосредоточено внимание на наиболее существенном, в кратких и четких формулировках обобщены важные теоретические положения.

Конспект представляет собой относительно подробное, последовательное изложение содержания прочитанного. На первых порах целесообразно в записях ближе держаться тексту, прибегая зачастую к прямому цитированию автора. В дальнейшем, по мере выработки навыков конспектирования, записи будут носить более свободный и сжатый характер.

Конспект книги обычно ведется в тетради. В самом начале конспекта указывается фамилия автора, полное название произведения, издательство, год и место издания. При цитировании обязательная ссылка на страницу книги. Если цитата взята из собрания сочинений, то необходимо указать соответствующий том. Следует помнить, что четкая ссылка на источник – непременное правило конспектирования. Если конспектируется статья, то указывается, где и когда она была напечатана.

Конспект подразделяется на части в соответсвтии с заранее продуманным планом. Пункты плана записываются в тексте или на полях конспекта. Писать его рекомендуется четко и разборчиво, так как небрежная запись с течением времени становиться малопонятной для ее автора. Существует правило: конспект, составленный для себя, должен быть по возможности написан так, чтобы его легко прочитал и кто-либо другой.

Формы конспекта могут быть разными и зависят от его целевого назначения (изучение материала в целом или под определенным углом зрения, подготовка к докладу, выступлению на занятии и т.д.), а также от характера произведения (монография, статья, документ и т.п.). Если речь идет просто об изложении содержания работы, текст конспекта может быть сплошным, с

выделением особо важных положений подчеркиванием или различными значками.

В случае, когда не ограничиваются переложением содержания, а фиксируют в конспекте и свои собственные суждения по данному вопросу или дополняют конспект соответствующими материалами их других источников, следует отводить место для такого рода записей. Рекомендуется разделить страницы тетради пополам по вертикали и в левой части вести конспект произведения, а в правой свои дополнительные записи, совмещая их по содержанию.

Конспектирование в большей мере, чем другие виды записей, помогает вырабатывать навыки правильного изложения в письменной форме важные теоретических и практических вопросов, умение четко их формулировать и ясно излагать своими словами.

Таким образом, составление конспекта требует вдумчивой работы, затраты времени и труда. Зато во время конспектирования приобретаются знания, создается фонд записей.

Конспект может быть текстуальным или тематическим. В текстуальном конспекте сохраняется логика и структура изучаемого произведения, а запись ведется в соответствии с расположением материала в книге. За основу тематического конспекта берется не план произведения, а содержание какой-либо темы или проблемы.

Текстуальный конспект желательно начинать после того, как вся книга прочитана и продумана, но это, к сожалению, не всегда возможно. В первую очередь необходимо составить план произведения письменно или мысленно, поскольку в соответствии с этим планом строится дальнейшая работа. Конспект включает в себя тезисы, которые составляют его основу. Но, в отличие от тезисов, конспект содержит краткую запись не только выводов, но и доказательств, вплоть до фактического материала. Иначе говоря, конспект – это расширенные тезисы, дополненные рассуждениями и доказательствами, мыслями и соображениями составителя записи.

Как правило, конспект включает в себя и выписки, но в него могут войти отдельные места, цитируемые дословно, а также факты, примеры, цифры, таблицы и схемы, взятые из книги. Следует помнить, что работа над конспектом только тогда будет творческой, когда она не ограничена текстом изучаемого произведения. Нужно дополнять конспект данными из другими источников.

В конспекте необходимо выделять отдельные места текста в зависимости от их значимости. Можно пользоваться различными способами: подчеркиваниями, вопросительными и восклицательными знаками, репликами, краткими оценками, писать на полях своих конспектов слова: «важно», «очень важно», «верно», «характерно».

В конспект могут помещаться диаграммы, схемы, таблицы, которые придадут ему наглядность.

Составлению тематического конспекта предшествует тщательное изучение всей литературы, подобранной для раскрытия данной темы. Бывает, что какая-либо тема рассматривается в нескольких главах или в разных местах книги. А в конспекте весь материал, относящийся к теме, будет сосредоточен в одном месте. В плане конспекта рекомендуется делать пометки, к каким источникам (вплоть до страницы) придется обратиться для раскрытия вопросов. Тематический конспект составляется обычно для того, чтобы глубже изучить определенный вопрос, подготовиться к докладу, лекции или выступлению на семинарском занятии. Такой конспект по содержанию приближается к реферату, докладу по избранной теме, особенно если включает и собственный вклад в изучение проблемы.

4 Подготовка к экзамену

Экзаменационная сессия – очень тяжелый период работы для студентов и ответственный труд для преподавателей. Главная задача экзаменов – проверка качества усвоения содержания дисциплины.

На основе такой проверки оценивается учебная работа не только студентов, но и преподавателей: по результатам экзаменов можно судить и о качестве всего учебного процесса. При подготовке к экзамену студенты повторяют материал курсов, которые они слушали и изучали в течение семестра, обобщают полученные знания, выделяют главное в предмете, воспроизводят общую картину для того, чтобы яснее понять связь между отдельными элементами дисциплины.

При подготовке к экзаменам основное направление дают программы курса и конспект, которые указывают, что в курсе наиболее важно. Основной материал должен прорабатываться по учебнику, поскольку конспекта недостаточно для изучения дисциплины. Учебник должен быть проработан в течение семестра, а перед экзаменом важно сосредоточить внимание на основных, наиболее сложных разделах. Подготовку по каждому разделу следует заканчивать восстановлением в памяти его краткого содержания в логической последовательности.

До экзамена обычно проводится консультация, но она не может возместить отсутствия систематической работы в течение семестра и помочь за несколько часов освоить материал, требующийся к экзамену. На консультации студент получает лишь ответы на трудные или оставшиеся неясными вопросы. Польза от консультации будет только в том случае, если студент до нее проработает весь материал. Надо учиться задавать вопросы, вырабатывать привычку пользоваться справочниками, энциклопедиями, а не быть на иждивении у преподавателей, который не всегда может тут же, «с ходу» назвать какой-либо факт, имя, событие. На экзамене нужно показать не только знание предмета, но и умение логически связно построить устный ответ.

Получив билет, надо вдуматься в поставленные вопросы для того, чтобы правильно понять их. Нередко студент отвечает не на тот вопрос, который поставлен, или в простом вопросе ищет скрытого смысла. Не поняв вопроса и не обдумав план ответа, не следует начинать писать. Конспект своего ответа надо рассматривать как план краткого сообщения на данную тему и составлять ответ нужно кратко. При этом необходимо показать умение выражать мысль четко и доходчиво.

Отвечать нужно спокойно, четко, продуманно, без торопливости, придерживаясь записи своего ответа. На экзаменах студент показывает не только свои знания, но и учится владеть собой. После ответа на билет могут следовать вопросы, которые имеют целью выяснить понимание других разделов курса, не вошедших в билет. Как правило, на них можно ответить кратко, достаточно показать знание сути вопроса. Часто студенты при ответе на дополнительные вопросы проявляют поспешность: не поняв смысла того, что у них спрашивают, начинают отвечать и нередко говорят не по сути.

Следует помнить, что необходимым условием правильного режима работы в период экзаменационной сессии является нормальный сон, поэтому подготовка к экзаменам не должна быть в ущерб сну. Установлено, что сильное эмоциональное напряжение во время экзаменов неблагоприятно отражается на нервной системе и многие студенты из-за волнений не спят ночи перед экзаменами. Обычно в сессию студенту не до болезни, так как весь организм озабочен одним - сдать экзамены. Но это еще не значит, что последствия неправильно организованного труда и чрезмерной занятости не скажутся потом. Поэтому каждый студент помнить о важности рационального распорядка рабочего дня и о своевременности снятия или уменьшения умственного напряжения.