

Министерство образования Республики Беларусь
Учреждение образования «Витебский государственный
университет имени П.М. Машерова»
Кафедра прикладного и системного программирования

А.И. Никитин

**СОВРЕМЕННЫЕ
ТЕХНОЛОГИИ РАЗРАБОТКИ
WEB-ПРИЛОЖЕНИЙ**

*Методические рекомендации
к выполнению лабораторных работ*

*Витебск
ВГУ имени П.М. Машерова
2023*

УДК 004.774(076.5)

ББК 32.973.43я73

Н62

Печатается по решению научно-методического совета учреждения образования «Витебский государственный университет имени П.М. Машерова». Протокол № 7 от 26.04.2023.

Автор: доцент кафедры прикладного и системного программирования ВГУ имени П.М. Машерова, кандидат физико-математических наук **А.И. Никитин**

Р е ц е н з е н т ы :

заведующий кафедрой информационных систем
и технологий УО «ВГТУ», кандидат технических наук,
доцент *В.Е. Казаков*;

старший преподаватель кафедры информационных технологий
и управления бизнесом ВГУ имени П.М. Машерова,
кандидат физико-математических наук, доцент *С.А. Шнаков*

Никитин, А.И.

Н62 Современные технологии разработки web-приложений :
методические рекомендации к выполнению лабораторных работ /
А.И. Никитин. – Витебск : ВГУ имени П.М. Машерова, 2023. – 44 с.

В методических рекомендациях показан ход выполнения лабораторных работ по различным темам, помогающим студентам освоить основы разметки гипертекста HTML, каскадных таблиц стилей CSS, языка программирования PHP. Содержатся задания для самостоятельной работы, практически по всем темам приводятся различные примеры, демонстрирующие те или иные возможности представленных технологий.

Предназначается для студентов специальности 1-40 05 01-07 Информационные системы и технологии (в здравоохранении).

УДК 004.774(076.5)

ББК 32.973.43я73

© Никитин А.И., 2023

© ВГУ имени П.М. Машерова, 2023

СОДЕРЖАНИЕ

Введение	4
Лабораторная работа № 1. Основы HTML и CSS	5
1.1. Ход лабораторной работы	5
1.1.1 Основы HTML	5
1.1.2 Основы CSS	8
1.1.3 Файловая структура	10
1.2. Самостоятельная работа «Языки программирования»	13
Лабораторная работа № 2. Основы блочной верстки	15
2.1. Ход лабораторной работы	15
2.2. Самостоятельная работа «Языки программирования»	20
Лабораторная работа № 3. Формы	21
3.1. Ход лабораторной работы	21
3.2. Самостоятельная работа «Работа с формами»	26
Лабораторная работа № 4. Общая характеристика языка программирования PHP	27
4.1. Ход лабораторной работы	27
4.2. Самостоятельная работа «Основы синтаксиса»	30
4.3. Самостоятельная работа «Массивы»	31
4.4. Самостоятельная работа «Строки»	33
Лабораторная работа № 5. Работа с формами	34
5.1. Ход лабораторной работы	34
5.2. Самостоятельная работа «Обработка формы»	35
5.3. Самостоятельная работа «Компьютерная игра»	35
Лабораторная работа № 6. Основы объектно-ориентированного программирования	36
6.1. Ход лабораторной работы	36
6.2. Самостоятельная работа «Классы и объекты»	40
6.3. Самостоятельная работа «Геометрические фигуры»	41
Список литературы	43

ВВЕДЕНИЕ

Быстрое развитие информационных технологий и глобальной сети Интернет привело к формированию информационной среды, оказывающей существенное влияние на все сферы человеческой деятельности. Корпоративные информационные системы стали одним из важнейших средств производства современной компании. Социальные сети позволяют обмениваться информацией миллионам пользователей вне зависимости от их географического местоположения. Всеобъемлющее распространение получил мобильный Интернет, с которым неразрывно связана не только повседневная жизнь, но и профессиональная деятельность подавляющего большинства людей по всему миру. В этих условиях умение создавать веб-приложения становится необходимой компетенцией дипломированного специалиста по информационным технологиям.

По охвату тем в методических рекомендациях рассматриваются начальные сведения о базовых основах разметки гипертекста HTML, каскадных таблицах стилей CSS. Более подробно рассматривается работа с языком программирования PHP и его применение для разработки веб-приложений. В частности, изучаются основы синтаксиса языка, основы работы с объектно-ориентированным подходом разработки веб-приложений.

ЛАБОРАТОРНАЯ РАБОТА № 1

ОСНОВЫ HTML И CSS

1.1. Ход лабораторной работы

1.1.1 Основы HTML

HTML (HyperText Markup Language, язык разметки гипертекста) – это система верстки, которая определяет, как и какие элементы должны располагаться на веб-странице. Информация на сайте, способ ее представления и оформления зависят исключительно от разработчика и тех целей, которые он перед собой ставит.

Для эффективной работы не обойтись без необходимых и привычных инструментов, в том числе и при написании кода HTML. Поэтому для начальной разработки веб-страниц рекомендуется использовать **VS Code**.

Чтобы браузер при отображении документа понимал, что имеет дело не с простым текстом, а с элементом форматирования и применяются теги. Общий синтаксис написания тегов следующий:

```
<тег атрибут1="значение" атрибут2="значение"/>
<тег атрибут1="значение" атрибут2="значение">...</тег>
```

Теги бывают двух типов – одиночные и парные. Одиночный тег используется самостоятельно, а парный может включать внутри себя другие теги или текст. У тегов допустимы различные атрибуты, которые разделяются между собой пробелом. Атрибуты можно подразделить на обязательные, они непременно должны присутствовать, и необязательные, их добавление зависит от цели применения тега.

Любая веб страница имеет следующую базовую структуру:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>My document</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

Элемент `<!DOCTYPE html>` – тип документа версии HTML5, тег `<html>` определяет начало HTML-файла, внутри него хранятся секции заголовка (`<head>`), где описываются настройки веб-страницы, и тела документа

(<body>), в котором содержится код веб-страницы, отображаемый пользователю. Тег <meta charset="UTF-8"> определяет кодировку веб-страницы UTF-8, тег <title> определяет заголовок вкладки окна браузера, тег <p> выводит абзац с текстом Hello world!.

Необходимо помнить, что несмотря на то, что отдельные теги выполняют функцию форматирования текста, передавая ему специфическое начертание, современный HTML является в первую очередь логической разметкой текста, то есть при разметке гипертекста веб-страниц в первую стоит выбирать определённый тег с точки зрения смысла текста, а только потом уже с точки зрения форматирования текста.

Для разметки текста можно использовать следующие теги:

<p> – абзац,
, – полужирный текст,
<i>, – курсивный текст,
<h1>, <h2>, <h3>, <h4>, <h5> – заголовки 1-5 уровней,
<q> – выделение в тексте цитат,
<blockquote> – выделение длинных цитат внутри документа в виде блока,
<cite> – выделения названий книг, журналов либо произведений.

Для формирования списка описаний или определения можно использовать следующую конструкцию:

```
<dl>
  <dt>HTML</dt>
  <dd>язык форматирования текста</dd>
</dl>
```

Для формирования списков можно использовать либо маркированный список:

```
<ul>
  <li>Пункт 1</li>
  <li>Пункт 2</li>
  <li>Пункт 3</li>
  <li>Пункт 4</li>
  <li>Пункт 5</li>
</ul>
```

либо нумерованный:

```
<ol>
  <li>Пункт</li>
  <li>Пункт</li>
  <li>Пункт</li>
  <li>Пункт</li>
  <li>Пункт</li>
</ol>
```

Для формирования гиперссылок использует тег `<a>` с обязательным атрибутом `href`:

```
<a href="https://google.com/" target="_blank">Ссылка на  
Google.com</a>
```

Данная ссылка будет открываться в отдельной вкладке браузера благодаря атрибуту `target` со значением `_blank`.

Вставка изображений осуществляется следующим способом:

```

```

Для разметки таблиц можно использовать следующие теги:

`<caption>` – название таблицы,

`<table>` – таблица,

`<th>` – ячейка заголовка,

`<tr>` – строка таблицы,

`<td>` – ячейка,

`<thead>` – шапка таблицы,

`<tbody>` – тело таблицы,

`<tfooter>` – подвал таблицы.

Разметка таблицы может выглядеть следующим образом:

```
<table>  
  <caption>Список студентов</caption>  
  <tr>  
    <th>№</th>  
    <th>ФИО</th>  
    <th>Математика</th>  
    <th>Физика</th>  
    <th>Русский язык</th>  
  </tr>  
  <tr>  
    <td>1</td>  
    <td>Петров</td>  
    <td>1</td>  
    <td>2</td>  
    <td>3</td>  
  </tr>  
  <tr>  
    <td>2</td>  
    <td>Петров</td>  
    <td>1</td>  
    <td>2</td>  
    <td>3</td>  
  </tr>  
</table>
```

1.1.2 Основы CSS

Стилем или CSS (Cascading Style Sheets, каскадные таблицы стилей) называется набор параметров форматирования, который применяется к элементам документа, чтобы изменить их внешний вид. Возможность работы со стилями издавна включают в развитые издательские системы и текстовые редакторы, тем самым позволяя одним нажатием кнопки придать тексту заданный, заранее установленный вид. Еще одним преимуществом стилей является то, что они предлагают намного больше возможностей для форматирования, чем обычный HTML.

Существует несколько способов подключения CSS к HTML.

Первый способ (самый распространённый) – подключение внешнего файла, путем добавления соответствующей разметки в <head>:

```
<link rel="stylesheet" href="style.css"/>
```

Второй способ (используется преимущественно различными фреймворками) – подключение стилей, используя тег <style>:

```
<style>
  body {
    color: red;
  }
</style>
```

Третий способ (используется при работа с Javascript) – подключение стилей напрямую для каждого тега:

```
<p style="color: red">Текст</p>
```

Как правило, стили размещаются в отдельном файле. Синтаксис описание стилей выглядит следующим образом:

```
td {
  background: olive;
  color: white;
  border: 1px solid black;
}
```

Для подключения стилей к конкретным тегам используют различные виды селекторов:

1. Селектор тегов – используется для определения базовых стилей.

```
p{
  font-weight: bold;
}
```

2. Селектор классов – основной тип селекторов, для стилизации одинаковых элементов.

```
.warning{  
    font-weight: bold;  
}
```

3. Селектор идентификаторов – для привязки стилей к конкретному тегу.

```
#warning{  
    font-weight: bold;  
}
```

4. Контекстный селектор – определяет стили элементов, которые должны лежать в других элементах.

```
p .warning{  
    font-weight: bold;  
}
```

5. Селектор соседних элементов – определяют стили элементов, которые стоят после определенного элемента.

```
b + i {  
    color: yellow;  
}
```

6. Селектор дочерних элементов – определяют стили, которые имеют конкретного прямого родителя.

```
p > i {  
    color: yellow;  
}
```

7. Селектор родственных элементов – похож на селектор соседних элементов, только элементы не обязательно должны идти друг за другом.

```
b ~ i {  
    color: yellow;  
}
```

8. Селектор атрибутов – привязывает стили к тегам с определённым значением атрибута тега (например, для стилизации тега <input>)

```
a[href="google.by"]{  
    font-weight: bold;  
}
```

9. Селектор псевдоэлементов – позволяют задать стиль элементов не определённых в дереве элементов документа, а также генерировать содержимое, которого нет в исходном коде текста (например, для создания различных 3D-эффектов или разнообразных анимаций).

```
p::first-letter{
    color: red;
    font-size: 60px;
}
```

10. Селектор псевдоклассов – определяют динамическое состояние элементов, которое изменяется с помощью действий пользователя, а также положение в дереве документа (например, наведение на ссылку, которая меняет свой цвет при наведении на неё курсора мыши).

```
p:hover{
    color: red;
}
```

11. Универсальный селектор – соответствует любому элементу веб-страницы.

```
{
    color: #000;
}
```

Как правило, основным типом селекторов для стилизации элементов являются – селекторы классов, остальные используются ситуативно и для решения задач, заточенных под конкретный тип селектора.

1.1.3 Файловая структура

Для создания простых веб-страниц как правило используется следующая структура проекта:

- index.html – главный html-файл;
- about.html (другие html-файлы);
- /css/ - директория для CSS-стилей;
- /img/ - директория для изображений;
- /js/ - директория для Javascript-файлов;
- /fonts/ - директория для подключаемых шрифтов.

Ознакомьтесь со следующим примером.

1. Создайте index.html со следующим кодом:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="./css/style.css">
</head>

<body>
  <h1>CSS</h1>
  <ol>
    <li><a href="#menu1">Способы подключения CSS к
документу</a></li>
    <li><a href="#menu2">Правила построения CSS</a></li>
    <li><a href="#menu3">Пример таблицы стилей</a></li>
  </ol>
  <p> <strong>CSS</strong> ( англ.<em>Cascading Style
Sheets</em> – каскадные таблицы стилей) – формальный язык
описания внешнего вида документа, написанного с использованием
языка разметки. Преимущественно используется как средство
описания, оформления внешнего вида веб-страниц, написанных с
помощью языков разметки HTML и XHTML, но может также
применяться к любым XML-документам, например, к SVG или
XUL.</p>
  <h2 id="menu1">Способы подключения CSS к документу</h2>
  <p>Правила CSS пишутся на формальном языке CSS и
располагаются в таблицах стилей, то есть таблицы стилей
содержат в себе правила CSS. Эти таблицы стилей могут
располагаться как в самом веб-документе, внешний вид которого
они описывают, так и в отдельных файлах, имеющих формат
CSS. (По сути, формат CSS – это обычный текстовый файл. В файле
.css не содержится ничего, кроме перечня правил CSS и
комментариев к ним.) есть, эти таблицы стилей могут быть
подключены, внедрены в описываемый ими веб-документ четырьмя
различными способами:
  <ul>
    <li>когда таблица стилей находится в отдельном файле,
она может быть подключена к веб-документу посредством тега
<span class="clgreen">&lt;link&gt;</span>, располагающегося в
этом документе между тегам &lt;head&gt; и &lt;/head&gt;. (Тег
<span class="clgreen">&lt;link&gt;</span> будет иметь атрибут
<span class="clblue">href</span>, имеющий значением адрес этой
таблицы стилей). Все правила этой таблицы действуют на
протяжении всего документа;
    </li>
```

``когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством директивы `@import`, располагающейся в этом документе между тегами `<style>` и `</style>` (которые, в свою очередь, располагаются в этом документе между тегами `<head>` и `</head>`;) сразу после тега `<style>`, которая также указывает (в своих скобках, после слова `url`) на адрес этой таблицы стилей. Все правила этой таблицы действуют на протяжении всего документа;

``

``когда таблица стилей описана в самом документе, она может располагаться в нём между тегами `<style>` и `</style>` (которые, в свою очередь, располагаются в этом документе между тегами `<head>` и `</head>`;) Все правила этой таблицы действуют на протяжении всего документа;

``

``когда таблица стилей описана в самом документе, она может располагаться в нём в теле какого-то отдельного тега (посредством его атрибута `style`) этого документа. Все правила этой таблицы действуют только на содержимое этого тега.

``

``

`</p>`

`<h2 id="menu2">Правила построения CSS</h2>`

`<p>`В первых трёх случаях подключения таблицы CSS к документу (см. выше) каждое правило CSS из таблицы стилей имеет две основные части – и блок объявлений. Селектор, расположенный в левой части правила, определяет, на какие части документа распространяется правило. Блок объявлений располагается в правой части правила. Он помещается в фигурные скобки, и, в свою очередь, состоит из одного или более объявлений, разделённых знаком «;». Каждое объявление представляет собой сочетание свойства CSS и значения, разделённых знаком ":". Селекторы могут группироваться в одной строке через запятую. В таком случае свойство применяется к каждому из них.

`</p>`

`<h2 id="menu3">Пример таблицы стилей</h2>`

`<p>`Пример таблицы стилей (в таком виде она может быть либо размещена в отдельном файле `.css` либо же обрамлена тегами `<style>` и размещена в «шапке» той самой веб-страницы, на которую она действует):`</p>`

`<pre>`

```
p {
    font-family: arial, helvetica, sans-serif;
}
h2 {
    font-size: 20pt;
    color: red;
    background: white;
}
```

```

        .note {
            color: red;
            background-color: yellow;
            font-weight: bold;
        }
        p#paragraph1 {
            padding-left: 10px;
        }
    </pre>
    <a href="#">Наверх</a>
</body>
</html>

```

2. Создайте style.css в директории /css/ со следующим кодом:

```

h1{
    font-size: 32px;
    font-weight: 700;
}
h2{
    font-size: 24px;
    font-style: italic;
}
p{
    font-size: 14px;
    font-weight: 400;
}
.clgreen{
    color: #0f0;
}
.clblue{
    color:blue;
}

```

3. Сохраните index.html и style.css и откройте html-файл в браузере, чтобы увидеть результат.

1.2. Самостоятельная работа «Языки программирования»

Ознакомиться с историей языка, согласно варианту (можно брать информацию с Википедии).

Варианты задания:

1. Java
2. Javascript
3. C++
4. C#

5. PHP
6. Python
7. Ruby
8. Golang
9. Visual Basic
10. C
11. SQL
12. Assembly

Подготовить небольшую статью с использованием HTML и CSS о данном языке согласно следующим критериям:

1. Текст должен состоять не менее чем из 3 разделов: история создания, краткий обзор языка, примеры кода.
2. В начале текста должно быть сформировано содержание в виде списка якорных ссылок на соответствующий раздел статьи.
3. Текст должен содержать как минимум 3 картинки с различным выравниванием (по левому краю, по правому краю и по центру) и обтеканием.
4. Текст должен содержать как минимум 1 таблицу. Форматирование должно быть выполнено по примеру.
5. Текст должен содержать 1 нумерованный список, 1 маркированный список со стандартным маркером, 1 маркированный список с маркером в виде картинки.
6. Размер шрифта заголовков 1 уровня – 24, заголовков 2 уровня – 18, основного текста – 14.
7. Для заголовков использовать нестандартный шрифт, подключенный локально через файл.
8. Текст должен содержать ссылки как внутренние, так и внешние.
9. Для форматирования текста использовать минимум 30 различных тегов и минимум 40 различных CSS-свойств.
10. В разделе "примеры кода" оформить сам код с цветовой подсветкой соответствующих элементов (по аналогии цветовых тем в различных IDE).
11. Перед началом работы не забудьте подключить `normalize.css` (<https://necolas.github.io/normalize.css/8.0.1/normalize.css>).
12. В конце статьи вывести список использованных тегов и CSS-свойств.

ЛАБОРАТОРНАЯ РАБОТА № 2 ОСНОВЫ БЛОЧНОЙ ВЕРСТКИ

2.1. Ход лабораторной работы

Как правило, веб-страница состоит из множества различных элементов, которые могут иметь сложную структуру. Поэтому при создании веб-страницы возникает необходимость нужным образом позиционировать эти элементы, стилизовать их так, чтобы они располагались на странице нужным образом. То есть возникает вопрос создания макета страницы, ее верстки.

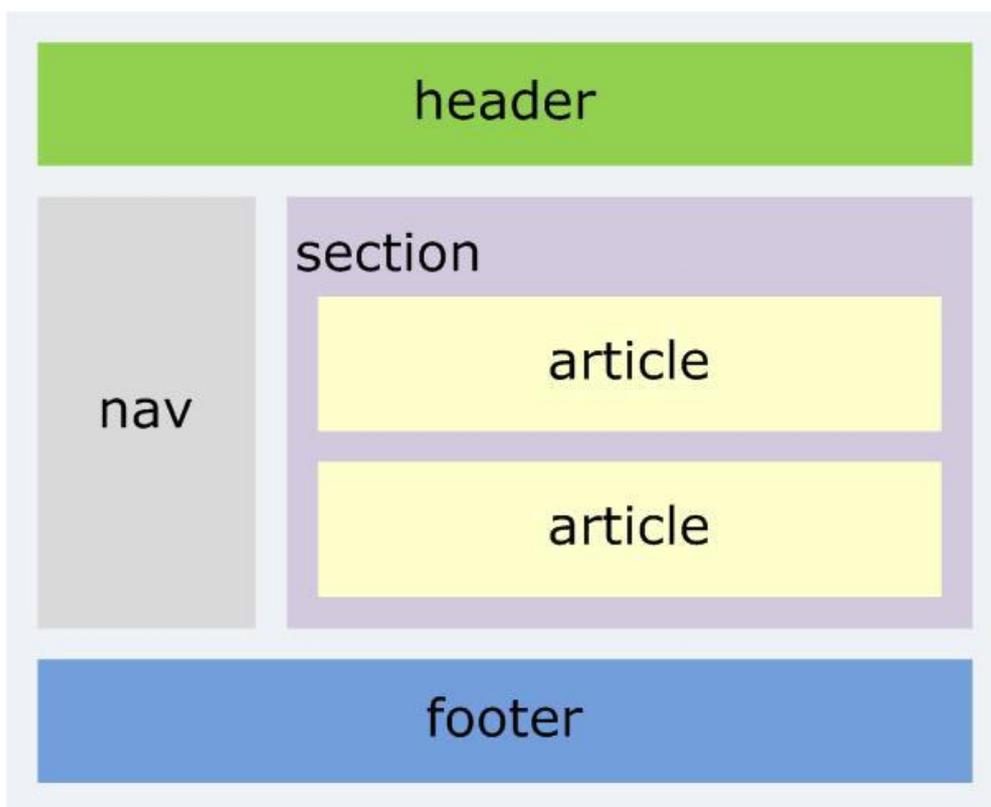


Рисунок 1. Блочная модель веб-страницы

Суть любой блочной модели CSS заключается в том, что в графическом редакторе создается обычный макет сайта, где разработчик размечает основные области: блок с главным контентом, боковую панель, низ («подвал»), шапка и так далее (Рис. 1). При этом он определяет размеры и форму, соотнося блоки в наиболее удобном для визуального восприятия виде. Чтобы в итоге получился отличный результат, принимают во внимание следующие принципы блочной верстки веб сайта:

1. **Принцип разделения кода.** Согласно этого правила, содержимое всегда следует отделять от непосредственного оформления. Другими словами, CSS и HTML делают в разных файлах, что делает структуру

понятней, а в работе - более удобной, поскольку позволяет разделить код, делая тем самым каждый из файлов более читаемым.

2. **Принцип использование тега div.** Как уже отмечалось выше, данный тег является не только универсальным, но и базовым элементом любой блочной структуры. Благодаря хорошему ранжированию поисковыми системами, такая страничка намного быстрее выводится в ТОП поисковых запросов. Однако не забывайте, что таблички, списки и другие специфичные элементы все равно лучше верстать с использованием специально-предназначенных тегов (table, ul).

Нужно понимать, что каждая система HTML элементов имеет блок, который непосредственно его окружает. При этом он всегда состоит из разных компонентов, управлять которыми можно независимо друг от друга за счет CSS, что очень удобно. Благодаря этому web-дизайнер имеет уникальную возможность свободно размещать элементы так, как это будет наиболее удобно для него. Все слои состоят из следующих компонентов (Рис. 2):

- **Высота.** Параметр, с помощью которого определяется высота элемента, причем она напрямую зависит от объема того контента, который содержится внутри.

- **Ширина.** Параметр, определяемый ширину содержимого внутреннего элемента важно отметить, что для всех блочных структур данное значение по умолчанию равняется 100%, однако разработчик может его изменить.

- **Border.** Элементы, которые ограничивают внутренние блоки. Они могут иметь самое разное оформление, цвет и структуру, а также быть даже невидимыми. Важно помнить, что границы объективно присутствуют в любой блочной верстке.

- **Margin.** Предназначены для того, чтобы определить расстояние между границами элемента, а также тем, что его непосредственно окружает.



Рисунок 2. Модель блочного элемента

Ширина блока формируется из ширины контента и значений полей, границ и отступов. В CSS3 есть свойство `box-sizing`, которое часто используется. При значении `border-box` ширина начинает включать поля и границы, но не отступы. Таким образом, подключая `box-sizing` со значением `border-box` к своему стилю, мы можем задавать ширину в процентах и спокойно указывать `border` и `padding`, не боясь, что изменится ширина блока.

Ознакомьтесь со следующим примером.

Файл **index.html**:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <link href="./css/style.css" rel="stylesheet">
    <title>Блочная верстка в HTML5</title>
  </head>
  <body>
    <div id="header">
      <h1>MySyte.com - Сайт о Lorem Ipsum</h1>
      <div id="nav">
        <ul>
          <li><a href="#">Главная</a></li>
          <li><a href="#">Блог</a></li>
          <li><a href="#">Форум</a></li>
          <li><a href="#">Контакты</a></li>
          <li><a href="#">О сайте</a></li>
        </ul>
      </div>
    </div>
    <div class="wrapper">
      <div id="sidebar1" class="aside">
        <h2>The standard Lorem Ipsum passage</h2>
        <p>"Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua..."</p>
      </div>
      <div id="sidebar2" class="aside">
        <h2>1914 translation by H. Rackham</h2>
        <p>It is a long established fact that a reader
will be distracted by the readable
content of a page when looking at its
layout.</p>
        <h3>Options</h3>
        <ul>
          <li>Item1</li>
          <li>Item2</li>
          <li>Item3</li>
        </ul>
      </div>
    </div>
  </body>
</html>
```

```

        </div>
        <div id="article">
            <h2>What is Lorem Ipsum?</h2>
            <p>Lorem Ipsum is simply dummy text of the
printing and typesetting industry...</p>
            <p>Contrary to popular belief, Lorem Ipsum is
not simply random text. It has roots in a piece of
            classical Latin literature from 45 BC, making
it over 2000 years old. Richard McClintock,
            a Latin professor at Hampden-Sydney College in
Virginia...</p>
        </div>
    </div>
    <div id="footer">
        <p>Contacts: admin@mysyte.com</p>
        <p>Copyright © MySyte.com, 2023</p>
    </div>
</body>
</html>

```

Файл style.css:

```

* {
    box-sizing: border-box;
}
html, body, div, span, h1, h2, h3, h4, h5, h6, p, a, ul, li{

    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    vertical-align: baseline;
}
body {
    font-family: Verdana, Arial, sans-serif;
    background-color: #f7f7f7;
}
#header{
    background-color: #f4f4f4;
}
#header h1 {
    font-size: 24px;
    text-transform: uppercase;
    font-weight: bold;
    padding: 30px 30px 30px 10px;
    clear: both;
}
#nav {
    background-color: #eee;
    border-top: 1px solid #ccc;
    border-bottom: 1px solid #ccc;
}

```

```

}
#nav li {
    float: left;
    list-style: none;
}
#nav a {
    display: block;
    color: black;
    padding: 10px 25px;
    text-decoration: none;
    border-right: 1px solid #ccc;
}
#nav li:last-child a {
    border-right: none;
}
#nav a:hover {
    font-weight: bold;
}
#nav:after {
    content: " ";
    display: table;
    clear: both;
}
.wrapper{
    background-color: #f7f7f7;
}
.aside h2 {
    font-size: 0.95em;
    margin-top: 15px;
}
.aside h3 {
    font-size: 0.85em;
    margin-top: 10px;
}
.aside p, .aside li {
    font-size: .75em;
    margin-top: 10px;
}
.aside li{
    list-style-type: none;
}
#sidebar1 {
    float: left;
    width: 20%;
    padding: 0 10px 0 20px;
}
#sidebar2 {
    float: right;
    width: 20%;
    padding: 0 20px 0 10px;
}

```

```

#article{
    background-color: #fafafa;
    border-left: 1px solid #ccc;
    border-right: 1px solid #ccc;
    margin-left: 20%;
    margin-right: 20%;
    padding: 15px;
    width: 60%;
}
#article:after{
    clear:both;
    display:table;
    content:'';
}
#article h2{
    font-size: 1.3em;
    margin-bottom:15px;
}
#article p{
    line-height: 150%;
    margin-bottom: 15px;
}
#footer{
    border-top: 1px solid #ccc;
    font-size: .8em;
    text-align: center;
    padding: 10px 10px 30px 10px;
}
#nav ul, #header h1, .wrapper, #footer p {
    max-width: 1200px;
    margin: 0 auto;
}
.wrapper, #nav, #header, #footer{
    min-width: 768px;
}

```

2.2. Самостоятельная работа «Языки программирования»

На основе Лабораторной работы 1 разработать блочную модель статьи о Вашем языке программирования (согласно варианту), в котором должны быть меню, контентная область, боковая модель, шапка и подвал.

ЛАБОРАТОРНАЯ РАБОТА № 3 ФОРМЫ

3.1. Ход лабораторной работы

Ознакомьтесь с примером реализации формы, с использованием блочной модели.

Файл **index.html**:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="./css/style1.css">
</head>
<body>
  <div class="container">
    <div class="row">
      <div class="col-lg-8">
        <form action="" method="post" name="reg_form" class="reg-
form">
          <div class="row">
            <div class="col">
              <div class="form-label">
                <label for="FIO">ФИО:</label>
              </div>
            </div>
            <div class="col">
              <div class="form-control">
                <input type="text" name="FIO" id="FIO" value=""
placeholder="Введите ФИО">
              </div>
            </div>
          </div>
          <div class="row">
            <div class="col">
              <div class="form-label">
                <label for="email">Email:</label>
              </div>
            </div>
            <div class="col">
              <div class="form-control">
                <input type="text" name="email" id="email" value=""
placeholder="Введите Email">
              </div>
            </div>
          </div>
        </form>
      </div>
    </div>
  </div>
</body>
</html>
```

```

    </div>
</div>
<div class="row">
  <div class="col">
    <div class="form-label">
      <label for="phone">Телефон:</label>
    </div>
  </div>
  <div class="col">
    <div class="form-control">
      <input type="text" name="phone" id="phone" value=""
placeholder="Введите телефон" readonly>
    </div>
  </div>
</div>
<div class="row">
  <div class="col">
    <div class="form-label">
      <label for="pass">Пароль:</label>
    </div>
  </div>
  <div class="col error">
    <div class="form-control">
      <input type="password" name="pass" id="pass" value=""
placeholder="Введите пароль">
    </div>
    <div class="error-text">
      Короткий пароль
    </div>
  </div>
</div>
<div class="row">
  <div class="col">
    <div class="form-label">
      <label for="dob">Дата рождения:</label>
    </div>
  </div>
  <div class="col">
    <div class="form-control">
      <input type="date" name="dob" id="dob" value=""
placeholder="Введите дату рождения">
    </div>
  </div>
</div>
<div class="row">
  <div class="col">
    <div class="form-label">
      <label>Хобби:</label>
    </div>
  </div>
  <div class="col">

```

```

    <div class="form-control">
      <div class="form-control-check">
        <input type="checkbox" name="hobby[]" id="hobby-1"
value="hobby-1"><label for="hobby-1">Спорт</label>
      </div>
      <div class="form-control-check">
        <input type="checkbox" name="hobby[]" id="hobby-2"
value="hobby-2"><label
        for="hobby-2">Программирование</label>
      </div>
      <div class="form-control-check">
        <input type="checkbox" name="hobby[]" id="hobby-3"
value="hobby-3"><label for="hobby-3">Рисование</label>
      </div>
    </div>
  </div>
</div>
<div class="row">
  <div class="col">
    <div class="form-label">
      <label>Образование:</label>
    </div>
  </div>
  <div class="col">
    <div class="form-control">
      <div class="form-control-radio">
        <input type="radio" name="education" id="education-1"
value="education-1"><label
        for="education-1">Базовое</label>
      </div>
      <div class="form-control-radio">
        <input type="radio" name="education" id="education-2"
value="education-2"><label
        for="education-2">Средне-специальное</label>
      </div>
      <div class="form-control-radio">
        <input type="radio" name="education" id="education-3"
value="education-3"><label
        for="education-3">Высшее</label>
      </div>
    </div>
  </div>
</div>
<div class="row">
  <div class="col">
    <div class="form-label">
      <label for="sex">Пол:</label>
    </div>
  </div>
  <div class="col">
    <div class="form-control">

```

```

        <select name="sex" id="sex">
            <option value="0">Выберите пол</option>
            <option value="f" selected>Женский</option>
            <option value="m" disabled>Мужской</option>
        </select>
    </div>
</div>
</div>
<div class="row">
    <div class="col">
        <div class="form-label">
            <label for="avatar">Аватар:</label>
        </div>
    </div>
    <div class="col">
        <div class="form-control">
            <input type="file" name="avatar" id="avatar" value="">
        </div>
    </div>
</div>
<div class="row">
    <div class="col">
    </div>
    <div class="col">
        <div class="form-control">
            <button>Отправить</button>
        </div>
    </div>
</div>
</form>
</div>
</div>
</div>
</body>
</html>

```

Файл style.css:

```

div{
    position: relative;
    float: left;
    box-sizing: border-box;
}
.container{
    width: 440px;
    margin: 0 auto;
    float: none;
    background: #fff;
}
.row{
    float: none;

```

```

}
.container:after, .container:before, .row:after, .row:before {
    display: table;
    content: " ";
    box-sizing: border-box;
    clear: both;
}
.col{
    width: 50%;
}
.reg-form{
    border: 1px solid #eee;
    position: relative;
    float: left;
    width: 100%;
    background: #f3f3f3;
    border-radius: 10px;
    box-shadow: 2px 2px 6px rgb(0 0 0 / 50%);
    box-sizing: border-box;
}
.form-label{
    float: right;
    padding: 15px 15px;
}
.form-label label{
}
}
.form-control{
    padding: 10px 15px;
    width: 75%;
}
}
.form-control-check, .form-control-radio{
    width: 100%;
}
}
.form-control-check:nth-child(1), .form-control-radio:first-
child{
    margin-top: 6px;
}
}
.form-control input[type="text"], .form-control
input[type="password"], .form-control input[type="date"],
select, .form-control input[type="file"]{
    width: 100%;
    padding: 5px 10px;
    border-radius: 5px;
    border: 1px solid #bfbfbf;
    box-sizing: border-box;
}
}
.error input[type="password"]{
    border-color: red;
}
}
.error-text{

```

```

    color: red;
    padding: 5px 25px;
    font-size: 12px;
}
.form-control button{
    width: 50%;
    padding: 5px 10px;
    border-radius: 5px;
    border: 1px solid #bfbfbf;
    box-sizing: border-box;
    background: #97f39f;
}
.form-control button:hover{
    background: green;
}
.form-control input[type="text"]:focus{
    border: 1px solid rgb(219, 89, 180);
}

```

3.2. Самостоятельная работа «Работа с формами»

Разработайте адаптивную форму согласно варианту, используя не менее **8 различных тегов** для создания форм и не менее **5 различных вариантов** атрибута **type** тега **input**. Адаптивный вариант подразумевает, что мобильная и десктопные версии будут отличаться. Так, например, подписи полей на десктопе делаются через тег *label* отдельно от **input**, а на мобильной версии подпись реализуется через атрибут тега **input - placeholder**, а тег *label* не отображается.

Варианты задания:

1. Полная форма регистрации пользователя в соц. сети.
2. Форма оформления заказа в интернет-магазине одежды.
3. Форма для соискателя вакансии на должность разработчика.
4. Формуляр читателя в библиотеке.
5. Форма для регистрации авто для продажи.
6. Форма для регистрации абитуриента при поступлении.
7. Форма заявления для регистрации брака.
8. Форма для подачи заявки на получение кредита в банке.
9. Форма для добавления товара в каталог.
10. Форма для подачи заявки на ремонт ноутбука.
11. Форма для заявки на получение визы.
12. Форма для покупки авиабилетов.

ЛАБОРАТОРНАЯ РАБОТА № 4

ОБЩАЯ ХАРАКТЕРИСТИКА ЯЗЫКА ПРОГРАММИРОВАНИЯ PHP

4.1. Ход лабораторной работы

Программа или скрипт на PHP, как правило, находится в файле расширением `.php`. Хотя разработчики могут также вставлять код `php` и в файлы с расширениями `.html/.htm`.

Документ PHP может содержать как разметку `html`, так и код на языке `php`. Для перехода от разметки `html` к коду `php` используются теги `<?php` и `?>`, между которыми идет код `php`. Данные теги служат указанием интерпретатору PHP, что их содержимое надо интерпретировать как код `php`, а не разметку `html`.

Ознакомьтесь с базовыми конструкциями языка.

1. Переменные

Переменные хранят отдельные значения, которые можно использовать в выражениях PHP. Для определения переменных применяется знак доллара `$`. Например:

```
$myVariable;
```

После объявления переменной вы можете присвоить ей значение:

```
$myVariable = 'Bob';
```

Вы можете сделать обе эти операции на одной и той же строке, если вы захотите:

```
$myVariable = 'Bob';
```

После установки значения переменной вы можете изменить его позже:

```
$myVariable = 'Bob';  
myVariable = 'Steve';
```

После определения переменной и присвоения ей значения мы можем использовать ее в выражениях PHP. Например, вывести ее значение на веб-страницу:

```
echo $myVariable;
```

Для вывода значения переменной применяется как правило команда `echo`, после которой указывалось выводимое значение. Однако есть и другой способ вывести значение переменной. Например, мы хотим одновременно вывести значения двух переменных:

```
$num_1 = 11;
```

```
$num_2 = 35;
```

```
echo "num_1 = $num_1 num_2=$num_2";
```

Переменные имеют разные типы данных:

Переменная	Пояснение	Пример
string	Последовательность текста, называемая строкой. Чтобы указать, что это значение является строкой, вы должны заключить его в кавычки.	<code>\$myVariable = 'Bob';</code>
int(float)	Числа. Числа не имеют кавычек вокруг них.	<code>\$myVariable = 10;</code>
bool	Значение True(Правда)/False(Ложь). Слова true и false специальные ключевые слова в PHP, и не нуждаются в кавычках.	<code>\$myVariable = true;</code>
array	Массив, который позволяет хранить несколько значений в одной ссылке.	<code>\$myVariable = [1,'Bob','Steve',10];</code> Обратиться к каждому элементу массива можно так: <code>\$myVariable[0]</code> , <code>\$myVariable[1]</code> , и т.д.
object	Экземпляр класса, созданный как правило с помощью конструктора класса.	<code>\$myVariable = new Person()</code> Обращение к полю: <code>\$myVariable.name</code>

2. Комментарии

Комментарии - это, по сути, короткие фрагменты текста, которые могут быть добавлены в код, и которые игнорируются браузером. Вы можете поместить комментарии в PHP, так же как вы делали это в CSS:

```
/*  
Всё, что находится тут - комментарий.  
*/
```

Если ваш комментарий не содержит переноса строк, то зачастую легче поставить две косые черты, как тут:

```
// Это комментарий
```

3. Операторы

Ниже приведены основные операторы, используемые в языке:

Оператор	Пояснение	Символ(ы)	Пример
Конкатенация	Склеивание двух строк вместе.	.	<code>echo "Hello"."world!";</code>
Сложение, Вычитание, Умножение, Деление	Они делают то, чего вы от них ожидаете в математике.	+,-, *, /	<code>9 + 1; 9 - 3; 8 * 2; 9 / 3;</code>

Оператор	Пояснение	Символ(ы)	Пример
Присваивание	присваивает значение переменной.	=	<code>\$myVariable = 'Bob';</code>
Равенство (Тождество)	Делает проверку, если увидит, что два значения равны друг другу, то возвращает результат true/false (Boolean).	==(===)	<code>\$myVariable = 3; \$myVariable == 4;</code>
Отрицание (Неравенство)	Возвращает логически противоположное значение, которое ему предшествует; превращает true в false, и т.д. Когда используется вместе с оператором равенства, оператор отрицания проверяет, являются ли два значения <i>не</i> равными.	!, !=, !==	Основное выражение true, но сравнение возвращает false, потому что мы отрицаем его: <code>\$myVariable = 3; !(\$myVariable === 3);</code> Здесь мы проверяем "\$myVariable НЕ равно 3". Это возвращает false, потому что \$myVariable равно 3. <code>\$myVariable = 3; \$myVariable !== 3;</code>

4. Условные операторы

Это конструкции в коде, которые позволяют проверить истинность или ложность выражения и выполнить другой код в зависимости от полученного результата. Самая распространённая форма условия — инструкция `if ... else`. Например:

```
$iceCream = 'chocolate';
if (iceCream === 'chocolate') {
    echo 'Yay, I love chocolate ice cream!';
} else {
    echo 'Awww, but chocolate is my favorite...';
}
```

Выражение внутри `if (...)` — это проверка, которая использует тождественный оператор (как описано выше), чтобы сравнить переменную `iceCream` со строкой `chocolate` и увидеть равны ли они. Если это сравнение возвращает `true`, выполнится первый блок кода. Если нет, этот код пропустится и выполнится второй блок кода, после инструкции `else`.

5. Операторы повторений

Для совершения повторяемых действий в РНР, как и в других языках программирования, используются циклы. В РНР имеются следующие виды циклов:

- `for`
- `while`

- do..while
- foreach (как правило для ассоциативных функций)

Пример цикла for:

```
for ($i = 1; $i < 10; $i++)
{
    echo "Квадрат числа $i равен " . $i * $i . "<br/>";
}
```

6. Функции

Функции – способ упаковки функциональности, которую вы хотите использовать повторно. Всякий раз, когда вам нужна определённая процедура, вы можете просто вызвать функцию по её имени, а не переписывать весь код каждый раз. Например:

```
function multiply(num1,num2) {
    var result = num1 * num2;
    return result;
}
```

Попробуйте запустить вышеупомянутую функцию в консоли, затем попробуйте изменить аргументы, например:

```
multiply(4,7);
multiply(20,20);
multiply(0.5,3);
```

4.2. Самостоятельная работа «Основы синтаксиса»

Напишите программу, исходя из Вашего варианта.

Варианты задания:

1. Дано число. Определить, является ли оно простым или нет.
2. Ряд Фибоначчи — это последовательность натуральных чисел, где каждое последующее число является суммой двух предыдущих: 1 1 2 3 5 8 13 21 34 55 89. Дано число N. Найти N-ый член последовательности Фибоначчи.
3. Бизнесмен взял ссуду M тысяч рублей в банке под 20% годовых. через сколько лет его долг превысит S тысяч рублей, если за это время он не будет отдавать долг.
4. Дан год. Определить, високосный это год или нет. Вывести в консоль соответствующую надпись, а также количество дней в году.
5. Дано число грибов. Вывести в консоль число и слово "грибы" с соответствующим окончанием (например, "17 грибов", "1 гриб", "2 гриба" и т.д.)

6. Дан день, месяц и год (3 числа). Определить, является ли дата корректной, не используя объект Date.

7. Дано число k (от 1 до 300) и последовательность следующего вида: 100101102103104105..198199. Вывести k -цифру в этой последовательности.

8. Пользователь задумал число от 1 до 100. Компьютер пытается его угадать (генерирует случайное число). После того, как компьютер называет число, пользователь говорит меньше, больше или равно загаданному. Компьютер соответственно генерирует заново число, но с учетом того, что сказал пользователь. Вывести все числа, которые сгенерировал компьютер перед тем как угадать число пользователя.

9. Дана дата в формате "день.месяц.год". Найти порядковый номер дня относительно начала года, не используя объект Date.

10. Дан прямоугольник размерами $N \times M$. Каждый раз от него отрезают квадрат со стороной равной меньшей стороне оставшегося прямоугольника. Например, прямоугольник 10×6 , то сначала отрезаю 6×6 , потом 4×4 и т.д. Сколько нужно сделать отрезаний до того, как получится квадрат.

11. Вычислить: $(1+2) \cdot (1+2+3) \cdot \dots \cdot (1+2+\dots+50)$.

12. Компания продает цветы с доставкой на дом. Компания работает с понедельника по субботу круглосуточно. Если клиент делает заказ с 8.00 до 16.00 в дни работы компании, то он может получить заказ после 16.00 в этот же день. Если же он делает заказ до 8.00 в дни работы компании, то заказ он сможет получить с 8.00 до 16.00 в этот же день. Если заказ сделан после 16.00, то доставка заказа можно будет получить на следующий рабочий день компании с 8.00 до 16.00. Заказы можно делать в любое время и любой день. Дан день недели и время заказа (часы). В какой ближайший интервал можно будет получить заказ (например, завтра с 8.00 до 16.00).

4.3. Самостоятельная работа «Массивы»

Напишите программу, обрабатывающую массив действительных чисел.

Варианты задания.

Часть А. Вычислите в массиве, не используя методов объекта Array:

1. сумму элементов массива от начала до первого элемента, удовлетворяющего условию: синус этого числа есть число положительное;
2. сумму элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа π не превосходит 10^{-5} ;
3. сумму элементов массива от последнего элемента, удовлетворяющего условию: синус этого числа есть число отрицательное — до конца массива;

4. сумму элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности квадратного и кубического корня из этого числа не превосходит 10^{-5} ;
5. сумму элементов массива от начала до последнего элемента, удовлетворяющего условию: косинус этого числа есть число отрицательное;
6. сумму элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа e не превосходит 10^{-5} ;
7. сумму элементов массива от первого элемента, удовлетворяющего условию: косинус этого числа есть число положительное – до конца массива;
8. произведение элементов массива от начала до первого элемента, удовлетворяющего условию: косинус этого числа есть число отрицательное;
9. произведение элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа e не превосходит 10^{-5} ;
10. произведение элементов массива от последнего элемента, удовлетворяющего условию: синус этого числа есть число отрицательное – до конца массива;
11. произведение элементов массива между первым и последним элементами, удовлетворяющими условию: модуль разности этого числа и числа π не превосходит 10^{-5} ;
12. произведение элементов массива от начала до последнего элемента, удовлетворяющего условию: косинус этого числа есть число положительное.

Часть Б. Используя методы объекта Array, удалите из массива все элементы, модуль целой части которых –

1. это число, все цифры которого чётные;
2. это число, все цифры которого нечётные;
3. это число, все цифры которого – простое числа;
4. это число, сумма цифр которого чётная;
5. это число, сумма цифр которого нечётная;
6. это число, сумма цифр которого – простое число;
7. это число, цифры которого образуют возрастающую арифметическую прогрессию;
8. это число, цифры которого образуют убывающую арифметическую прогрессию;
9. это число, цифры которого образуют возрастающую геометрическую прогрессию;
10. это число, цифры которого образуют убывающую геометрическую прогрессию;

11. это число, у которого сумма цифр чётных разрядов, равна сумме цифр нечётных разрядов;
12. это число, которое при перестановке его цифр в обратном порядке не изменяется.

4.4. Самостоятельная работа «Строки»

Напишите программу, выполняющую обработку строк.

Варианты задания:

1. Дана строка. Подсчитать количество слогов в слове.
2. Дана строка. Получить строку из данной строки, где нечетные буквы будут заглавными, а четные - строчными.
3. Дано полное имя файла. Вывести все названия папок, которые входят в заданный путь.
4. Дана строка. Поделить строку на фрагменты по 3 символа в каждой.
5. Дано слово. Проверить, является ли данное слово палиндромом.
6. Дано предложение. Найти самое длинное слово в предложении.
7. Дан массив из слов. Проверить условие, что первая буква каждого слова равна последней букве предыдущего слова.
8. Дано 2 предложения. Посчитать количество слов, которые есть в обоих предложениях.
9. Дана строка, состоящая из слов, между которыми может быть больше одного пробела. Удалить все лишние пробелы, т.е. оставить по 1 пробелу между словами.
10. Дано предложение на русском языке. Определить, является ли оно панграммой.
11. Даны 2 слова. Проверить, состоят ли они из одних и тех же букв с учетом повторений и без учета регистра.
12. Дана строка. Вывести символы, которые в ней повторяются.

ЛАБОРАТОРНАЯ РАБОТА № 5 РАБОТА С ФОРМАМИ

5.1. Ход лабораторной работы

Одним из основных способов передачи данных веб-сайту является обработка форм. Формы представляют специальные элементы разметки HTML, которые содержат в себе различные элементы ввода - текстовые поля, кнопки и т.д. И с помощью данных форм мы можем ввести некоторые данные и отправить их на сервер. А сервер уже обрабатывает эти данные.

Создание форм состоит из следующих аспектов:

1. Создание элемента `<form>` в разметке HTML.
2. Добавление в этот элемент одно или несколько полей ввода.
3. Установка метода передачи данных. Чаще всего используются методы GET или POST.
4. Установка адреса, на который будут отправляться введенные данные.

Создадим форму. Для этого определим новый файл `form.php`, в который поместим следующее содержимое:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Форма</title>
</head>
<body>
  <h3>Форма ввода данных</h3>
  <form action="user.php" method="POST">
    <p>Имя: <input type="text" name="name" /></p>
    <p>Возраст: <input type="number" name="age" /></p>
    <input type="submit" value="Отправить">
  </form>
</body>
</html>
```

Теперь определим файл `user.php`, который будет иметь следующее содержание:

```
<?php
$name = "не определено";
$age = "не определен";
if(isset($_POST["name"])){

    $name = $_POST["name"];
```

```

}
if(isset($_POST["age"])){
    $age = $_POST["age"];
}
echo "Имя: $name <br> Возраст: $age";
?>

```

Для обработки запросов типа POST в PHP используется встроенная глобальная переменная `$_POST`. Она представляет ассоциативный массив данных, переданных с помощью метода POST. Используя ключи, мы можем получить отправленные значения. Ключами в этом массиве являются значения атрибутов `name` у полей ввода формы.

Например, так как атрибут `name` поля ввода возраста имеет значение `age` (`<input type="number" name="age" />`), то в массиве `$_POST` значение этого поля будет представлять ключ `"age": $_POST["age"]`.

И поскольку возможны ситуации, когда поле ввода будет не установлено, то в этом случае желательно перед обработкой данных проверять их наличие с помощью функции `isset()`. И если переменная установлена, то функция `isset()` возвратит значение `true`.

5.2. Самостоятельная работа «Обработка формы»

Доработать форму из Лабораторной работы 3. Добавить обработку ошибок при вводе данных в форму: проверка на заполнение обязательных полей формы, проверка на корректность значений, введенных в поля. В случае успешного заполнения формы вывести данные на экран. В случае если форма содержит ошибки, то вывести форму с уже введенными данными и отобразить список ошибок, допущенных при выводе формы.

5.3. Самостоятельная работа «Компьютерная игра»

Разработайте приложение с графическим интерфейсом, где должна быть реализована игра на одном экране для двух игроков. Реализацию ходов можно реализовывать через ввод поля формы.

Варианты задания

- | | |
|--|--------------------------|
| 1. Шахматы. | 8. Монополия (Менеджер). |
| 2. Шашки. | 9. Ромб. |
| 3. ГО. | 10. Балда. |
| 4. Морской бой. | 11. Реверси. |
| 5. Домино. | 12. Мельница. |
| 6. Нарды. | |
| 7. Настольная игра с кубиками и фишками. | |

ЛАБОРАТОРНАЯ РАБОТА № 6 ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

6.1. Ход лабораторной работы

При создании программы на PHP и отдельных ее блоков вполне может хватить той функциональности, которую представляют функции. Однако PHP имеет и другие возможности по созданию программ, которые представляет объектно-ориентированное программирование. В ряде случаев программы, использующие ООП, проще в понимании, их легче поддерживать и изменять.

Ключевыми понятиями парадигмы ООП являются понятия "класс" и "объект". Описанием объекта является класс, а объект представляет экземпляр этого класса. Для создания класса в PHP используется ключевое слово `class`, после которого идет название класса и фигурные скобки `{}` – блок кода класса. Например, новый класс, представляющий пользователя:

```
class Person
{
    public $name = "Undefined", $age = 18;

    function hello()
    {
        echo "Hello!<br>";
    }
}
```

Класс может содержать переменные, которые описывают какие-то признаки объекта, его состояние и которые еще называют свойствами или атрибутам. И также класс может содержать функции, которые еще называют методами и которые определяют его поведение.

Для обращения к свойствам и методам объекта внутри его класса применяется ключевое слово `$this`. Для обращения к полям и методам внутри класса также применяется оператор доступа `->`, перед которым идет `$this`. Причем это `$this` указывает именно на текущий объект.

```
function displayInfo()
{
    echo "Name: ".$this->name ."; Age: ".$this->age;
}
```

Обращаться к полям и методам объекта можно следующим образом:

```
$tom = new Person();
$tom->name = "Tom"; // установка свойства $name
$tom->age = 36; // установка свойства $age
$personName = $tom->name; // получение значения свойства $name
```

Конструкторы представляют специальные методы, которые выполняются при создании объекта и служат для начальной инициализации его свойств. Для создания конструктора надо объявить функцию с именем `__construct` (с двумя подчеркиваниями впереди):

```
<?php
class Person
{
    public $name, $age;
    function __construct($name, $age)
    {
        $this->name = $name;
        $this->age = $age;
    }

    function displayInfo()
    {
        echo "Name: $this->name; Age: $this->age<br>";
    }
}
$tom = new Person("Tom", 36);
$tom -> displayInfo();

$bob = new Person("Bob", 41);
$bob -> displayInfo();
?>
```

Наследование является одним из основных аспектов объектно-ориентированного программирования. Наследование позволяет классу взять функционал уже имеющихся классов и при необходимости переопределить его. Если у нас есть какой-нибудь класс, в котором не хватает пары функций, то гораздо проще переопределить имеющийся класс, написав пару строк, чем создавать новый с нуля, переписывая кучу кода.

Чтобы наследовать один класс от другого, нам надо применить оператор `extends`. Стоит отметить, что в PHP мы можем унаследовать класс только от одного класса. Множественное наследование не поддерживается.

Например, унаследуем класс `Employee` от класса `Person`:

```
<?php
class Person
{
    public $name;
    function __construct($name)
    {
        $this->name = $name;
    }
    function displayInfo()
    {
        echo "Имя: $this->name<br>";
    }
}
```

```

class Employee extends Person
{

$tom = new Employee("Tom");
$tom -> displayInfo();
?>

```

В данном случае предположим, что класс `Person` представляет человека в целом, а класс `Employee` – работника некоего предприятия. В этой связи каждый работник представляет человека. И чтобы не дублировать один и тот же функционал, лучше в данном случае унаследовать класс работника – `Employee` от класса человека – `Person`. В этой паре класс `Person` еще называется родительским или базовым классом, а класс – `Employee` – производным классом или классом-наследником.

С помощью специальных модификаторов можно задать область видимости для свойств и методов класса. В PHP есть три таких модификатора:

public: к свойствам и методам, объявленным с данным модификатором, можно обращаться из внешнего кода и из любой части программы;

protected: свойства и методы с данным модификатором доступны из текущего класса, а также из классов-наследников;

private: свойства и методы с данным модификатором доступны только из текущего класса.

Рассмотрим на примере:

```

class Person
{
    private $privateA = "private";
    public $publicA = "public";
    protected $protectedA = "protected";
    private function getPrivateMethod()
    {
        echo "private method <br />";
    }

    protected function getProtectedMethod()
    {
        echo "protected method <br />";
    }
    public function getPublicMethod()
    {
        echo "public method <br />";
    }
    function test()
    {
        $this->getPrivateMethod();
    }
}

```

```

    $this->getProtectedMethod();
    $this->getPublicMethod();

    echo "$this->privateA <br />";
    echo "$this->protectedA <br />";
    echo "$this->publicA <br />";
}
}

```

Класс определяет три свойства и три метода с разными модификаторами доступа. Из любого метода этого класса мы можем обратиться к любому методу и любому свойству. Если метод не имеет модификатора доступа, то по умолчанию его видимость аналогична модификатору `public`.

Кроме обычных методов и свойств с разными модификаторами доступа класс может содержать статические методы и свойства. Такие методы и свойства помечаются ключевым словом `static`. Статические методы и свойства создаются один раз для всего класса и относятся ко всему классу, тогда для нестатических свойств и методов создается отдельная копия для каждого объекта.

```

<?php
class Person
{
    public $name, $age;
    static $retirementAge = 65;
    function __construct($name, $age)
    {
        $this->name = $name;
        $this->age = $age;
    }
    function sayHello()
    {
        echo "Привет, меня зовут $this->name<br>";
    }
    static function printPerson($person)
    {
        echo "Имя: $person->name Возраст: $person->age<br>";
    }
}
?>

```

При обращении к статическим методам и свойствам используется имя класса и оператор `::`, вместо операции доступа `->`:

```

Person::printPerson($tom);
echo "Пенсионный возраст: " . Person::$retirementAge .
"<br>";

```

6.2. Самостоятельная работа «Классы и объекты»

Разработать классы для описанных ниже объектов по вариантам. Включить в класс методы set (...), get (...), show (...). Определить другие методы.

1. Student: Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс. Создать массив объектов. Вывести:

- а) список студентов заданного факультета;
- б) списки студентов для каждого факультета и курса;
- в) список студентов, родившихся после заданного года.

2. Abiturient: Фамилия, Имя, Отчество, Адрес, Оценки. Создать массив объектов. Вывести:

- а) список абитуриентов, имеющих неудовлетворительные оценки;
- б) список абитуриентов, сумма баллов у которых не меньше заданной;
- в) выбрать N абитуриентов, имеющих самую высокую сумму баллов, и список абитуриентов, имеющих полупроходной балл.

3. Aeroflot: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Создать массив объектов. Вывести:

- а) список рейсов для заданного пункта назначения;
- б) список рейсов для заданного дня недели;
- в) список рейсов для заданного дня недели, время вылета для которых больше заданного.

4. Book: Автор, Название, Издательство, Год, Количество страниц. Создать массив объектов. Вывести:

- а) список книг заданного автора;
- б) список книг, выпущенных заданным издательством;
- в) список книг, выпущенных после заданного года.

5. Worker: Фамилия и инициалы, Должность, Год поступления на работу, Зарплата. Создать массив объектов. Вывести:

- а) список работников, стаж работы которых на данном предприятии превышает заданное число лет;
- б) список работников, зарплата которых больше заданной;
- в) список работников, занимающих заданную должность.

6. Train: Пункт назначения, Номер поезда, Время отправления, Число общих мест, Купейных, Плацкартных. Создать массив объектов. Вывести:

- а) список поездов, следующих до заданного пункта назначения;

б) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

в) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

7. Product: Наименование, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести:

а) список товаров для заданного наименования;

б) список товаров для заданного наименования, цена которых не превышает указанной;

в) список товаров, срок хранения которых больше заданного.

8. Patient: Фамилия, Имя, Отчество, Адрес, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести:

а) список пациентов, имеющих данный диагноз;

б) список пациентов, номер медицинской карты которых находится в заданном интервале.

9. Bus: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. Создать массив объектов. Вывести:

а) список автобусов для заданного номера маршрута;

б) список автобусов, которые эксплуатируются больше 10 лет;

в) список автобусов, пробег у которых больше 10 000 км.

10. Phone: Фамилия, Имя, Отчество, Адрес, Номер, Время внутригородских разговоров, Время междугородних разговоров. Создать массив объектов. Вывести:

а) сведения об абонентах, время внутригородских разговоров которых превышает заданное;

б) сведения об абонентах, воспользовавшихся междугородней связью;

в) сведения об абонентах, выведенные в алфавитном порядке.

6.3. Самостоятельная работа «Геометрические фигуры»

Разработать программу, реализующую иерархию из 2 классов, включающих функции для ввода-вывода параметров объектов и некоторую заданную функцию. Продемонстрировать работу с каждым классом:

1. Базовый класс – круг, производный класс – кольцо, функция – площадь.

2. Базовый класс – сфера, производный класс – цилиндр, функция – объем.

3. Базовый класс – треугольник, производный класс – трапеция, функция – площадь.
4. Базовый класс – цилиндр, производный класс – конус, функция – объем.
5. Базовый класс – сфера, производный класс – цилиндр, функция – площадь поверхности.
6. Базовый класс – круг, производный класс – сектор, функция – площадь.
7. Базовый класс – куб, производный класс – прямоугольный параллелепипед, функция – объем.
8. Базовый класс – прямоугольник, производный класс – трапеция, функция – площадь.
9. Базовый класс – сфера, производный класс – конус, функция – объем.
10. Базовый класс – цилиндр, производный класс – конус, функция – площадь поверхности.

СПИСОК ЛИТЕРАТУРЫ

1. Робсон, Э. Изучаем HTML, XHTML и CSS / Э. Робсон, Э. Фримен, В. Черник; [пер. с англ. В. Черник]. – 2-е изд. – Санкт-Петербург [и др.]: Питер, 2022. – 718, [2] с.: ил. – (Head First O`Reilly). – ISBN 978-0596159900. – ISBN 978-5-4461-1247-0.
2. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон, Н. Вильчинский; [пер. с англ. Н. Вильчинский]. – 5-е изд. – Санкт-Петербург [и др.]: Питер, 2022. – 815, [1] с.: ил. – (Бестселлеры O`Reilly). – ISBN 978-1491978917. – ISBN 978-5-4461-0825-1.
3. Роббинс, Д. HTML5: карманный справочник, 5-е издание / пер. с англ. – М.: ООО «И.Д. Вильямс», 2015. – 192 с.
4. Грант, К. CSS для профи. – СПб.: Питер, 2019. – 496 с.
5. Макфарланд, Д. Новая большая книга CSS. – СПб.: Питер, 2016. – 720 с.
6. Мейер, Э. CSS: полный справочник, 4-е издание / Э. Мейер, Э. Уэйл // пер. с англ. – СПб.: ООО «Диалектика», 2019. – 1088 с.
7. Савельева, Н.В. Web-программирование: учеб.-метод. комплекс для студентов физ.-мат. фак. спец. 1-31 03 03-01 Прикладная математика / Н.В. Савельева, А.А. Трофимук; УО «Брестский гос. ун-т имени А.С. Пушкина». – Брест: БрГУ имени А. С. Пушкина, 2013. – 171 с.

Учебное издание

НИКИТИН Александр Игоревич

**СОВРЕМЕННЫЕ ТЕХНОЛОГИИ
РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЙ**

Методические рекомендации
к выполнению лабораторных работ

Технический редактор

Г.В. Разбоева

Компьютерный дизайн

Л.В. Рудницкая

Подписано в печать 14.11.2023. Формат 60x84 ¹/₁₆. Бумага офсетная.

Усл. печ. л. 2,56. Уч.-изд. л. 1,50. Тираж 25 экз. Заказ 134.

Издатель и полиграфическое исполнение – учреждение образования
«Витебский государственный университет имени П.М. Машерова».

Свидетельство о государственной регистрации в качестве издателя,
изготовителя, распространителя печатных изданий

№ 1/255 от 31.03.2014.

Отпечатано на ризографе учреждения образования
«Витебский государственный университет имени П.М. Машерова».

210038, г. Витебск, Московский проспект, 33.