Лабораторная работа №2.

Передача значений переменным в сценариях PHP

Цель: Получение практических навыков по передаче значений переменным в сценариях PHP с использованием формы.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Основы клиент-серверных технологий

PHP – это скриптовый язык, обрабатываемый сервером. Уточним, что такое сервер, какие функции он выполняет и какие вообще бывают серверы. Понятие сервер неразрывно связано с понятием клиент. Объединяет их компьютерная архитектура клиент-сервер. Обычно, когда говорят «сервер», имеют в виду сервер в архитектуре клиент-сервер, а когда говорят «клиент» – имеют в виду клиент в этой же архитектуре. Суть этой архитектуры в том, чтобы разделить функции между двумя подсистемами: клиентом, который отправляет запрос на выполнение каких-либо действий, и сервером, который выполняет этот запрос. Взаимодействие между клиентом и сервером происходит посредством стандартных специальных протоколов, таких как TCP/IP. На самом деле протоколов очень много, они различаются по уровням.

Сервер представляет собой набор программ, которые контролируют выполнение различных процессов. Соответственно, этот набор программ установлен на каком-то компьютере. Часто компьютер, на котором установлен сервер, и называют сервером. Основная функция компьютера-сервера – по запросу клиента запустить какой-либо определенный процесс и отправить клиенту результаты его работы.

Клиентом называют любой процесс, который пользуется услугами сервера. Клиентом может быть как пользователь, так и программа. Основная задача клиента – выполнение приложения и осуществление связи с сервером, когда этого требует приложение. То есть клиент должен предоставлять

пользователю интерфейс для работы с приложением, реализовывать логику его работы и при необходимости отправлять задания серверу.

Взаимодействие между клиентом и сервером начинается по инициативе клиента. Клиент запрашивает вид обслуживания, устанавливает сеанс, получает нужные ему результаты и сообщает об окончании работы.

Услугами одного сервера чаще всего пользуется несколько клиентов одновременно. Поэтому каждый сервер должен иметь достаточно большую производительность и обеспечивать безопасность данных.

Логичнее всего устанавливать сервер на компьютере, входящем в какую-либо сеть, локальную или глобальную. Однако можно устанавливать сервер и на отдельно стоящий компьютер (тогда он будет являться одновременно и клиентом и сервером).

Существует множество типов серверов. Вот лишь некоторые из них.

- **Видеосервер** сервер специально приспособлен к обработке изображений, хранению видеоматериалов, видеоигр и т.п. В связи с этим компьютер, на котором установлен видеосервер, должен иметь высокую производительность и большую память.
- **Поисковый** сервер предназначен для поиска информации в Internet.
- **Почтовый** сервер предоставляет услуги в ответ на запросы, присланные по электронной почте.
- **Сервер WWW** предназначен для работы в Internet.
- **Сервер баз данных** выполняет обработку запросов к базам данных.
- Сервер защиты данных предназначен для обеспечения безопасности данных (содержит, например, средства для идентификации паролей).
- **Сервер приложений** предназначен для выполнения прикладных процессов. С одной стороны взаимодействует с клиентами, получая задания, а с другой – работает с базами данных, подбирая необходимые для обработки данные.
- **Сервер удаленного доступа** обеспечивает коллективный удаленный доступ к данным.

- **Файловый сервер** обеспечивает функционирование распределенных ресурсов, предоставляет услуги поиска, хранения, архивирования данных и возможность одновременного доступа к ним нескольких пользователей.

Обычно на компьютере-сервере работает сразу несколько программ-серверов. Одна занимается электронной почтой, другая распределением файлов, третья предоставляет web-страницы.

Из всех типов серверов нас в основном интересует сервер WWW. Часто его называют web-сервером, http-сервером или даже просто сервером. Что представляет собой web-сервер? Во-первых, это хранилище информационных ресурсов. Во-вторых, эти ресурсы хранятся и предоставляются пользователям в соответствии со стандартами Internet (такими, как протокол передачи данных HTTP). Работа с документами web-сервера осуществляется при помощи браузера (например, IE, Opera или Mozilla), который отсылает серверу запросы, созданные в соответствии с протоколом HTTP. В процессе выполнения задания сервер может связываться с другими серверами.

Протокол HTTP и способы передачи данных на сервер

Internet построен по многоуровневому принципу, от физического уровня, связанного с физическими аспектами передачи двоичной информации, и до прикладного уровня, обеспечивающего интерфейс между пользователем и сетью.

HTTP (HyperText Transfer Protocol, протокол передачи гипертекста) – это протокол прикладного уровня, разработанный для обмена гипертекстовой информацией в Internet.

HTTP предоставляет набор методов для указания целей запроса, отправляемого серверу. Эти методы основаны на дисциплине ссылок, где для указания ресурса, к которому должен быть применен данный метод, используется универсальный идентификатор ресурсов (Universal Resource

Identifier) в виде местонахождения ресурса (Universal Resource Locator, URL) или в виде его универсального имени (Universal Resource Name, URN).

Протокол реализует принцип запрос/ответ. Запрашивающая программа – клиент инициирует взаимодействие с отвечающей программой – сервером и посылает запрос, содержащий:

- метод доступа;
- адрес URL;
- версию протокола;
- сообщение с информацией о типе передаваемых данных, информацией о клиенте, пославшем запрос, и, возможно, с содержательной частью (телом) сообщения.

Ответ сервера содержит:

- строку состояния, в которую входит версия протокола и код возврата (успех или ошибка);
- сообщение, в которое входит информация сервера, метаинформация (т.е. информация о содержании сообщения) и тело сообщения.

Использование HTML-форм для передачи данных на сервер

Для передачи данных на сервер в языке HTML есть специальная конструкция – формы. Формы предназначены для того чтобы получать от пользователя информацию. Например, вам нужно знать логин и пароль пользователя для того, чтобы определить, на какие страницы сайта его можно допускать. Или вам необходимы личные данные пользователя, чтобы была возможность с ним связаться. Формы как раз и применяются для ввода такой информации. В них можно вводить текст или выбирать подходящие варианты из списка. Данные, записанные в форму, отправляются для обработки специальной программе (например, скрипту на PHP) на сервере. В зависимости от введенных пользователем данных эта программа может формировать различные web-страницы, отправлять запросы к базе данных, запускать различные приложения и т.п.

Для создания формы в языке HTML используется тег FORM. Внутри него находится одна или несколько команд INPUT. С помощью атрибутов ACTION и METHOD тега FORM задаются имя программы, которая будет обрабатывать данные формы, и метод запроса, соответственно. Команда INPUT определяет тип и различные характеристики запрашиваемой информации. Отправка данных формы происходит после нажатия кнопки INPUT типа SUBMIT

<form action="1.php" method=POST> …… …… …… </form>	Данные формы будет обрабатывать файл 1.php, при отправке запроса будет использован метод POST.

При отправке данных формы с помощью метода GET содержимое формы добавляется к URL после знака вопроса в виде пар имя=значения, объединенных с помощью амперсанта &:

action?name1=value1&name2=value2&name3=value3

Здесь action – это URL-адрес программы, которая должна обрабатывать форму (это либо программа, заданная в атрибуте action тега form, либо сама текущая программа, если этот атрибут опущен). Имена name1, name2, name3 соответствуют именам элементов формы, а value1, value2, value3 – значениям этих элементов. Все специальные символы, включая = и &, в именах или значениях этих параметров будут опущены. Поэтому не стоит использовать в названиях или значениях элементов формы эти символы и символы кириллицы в идентификаторах.

Для кнопок типа checkbox и radiobutton значение value определяется атрибутом VALUE в том случае, когда кнопка отмечена. Не отмеченные кнопки при составлении строки запроса игнорируются целиком. Несколько кнопок типа checkbox могут иметь один атрибут NAME (и различные VALUE), если это необходимо. Кнопки типа radiobutton предназначены для одного из всех предложенных вариантов и поэтому должны иметь одинаковый атрибут NAME и различные атрибуты VALUE.

Для метода POST содержимое формы кодируется точно так же, как для метода GET, но вместо добавления строки к URL содержимое запроса посылается блоком данных как часть операции POST. Если присутствует атрибут ACTION, то значение URL, которое там находится, определяет, куда посылать этот блок данных. Этот метод рекомендуется для передачи больших по объему блоков данных.

Передать данные методом POST можно только с помощью HTML-формы, поскольку данные передаются в теле запроса, а не в заголовке, как в GET. Соответственно и изменить значение параметров можно, только изменив значение, введенное в форму. При использовании POST пользователь не видит передаваемые серверу данные.

Основное преимущество POST запросов – это их большая безопасность и функциональность по сравнению с GET-запросами. Поэтому метод POST чаще используют для передачи важной информации, а также информации большого объема.

ПРАКТИЧЕСКАЯ ЧАСТЬ

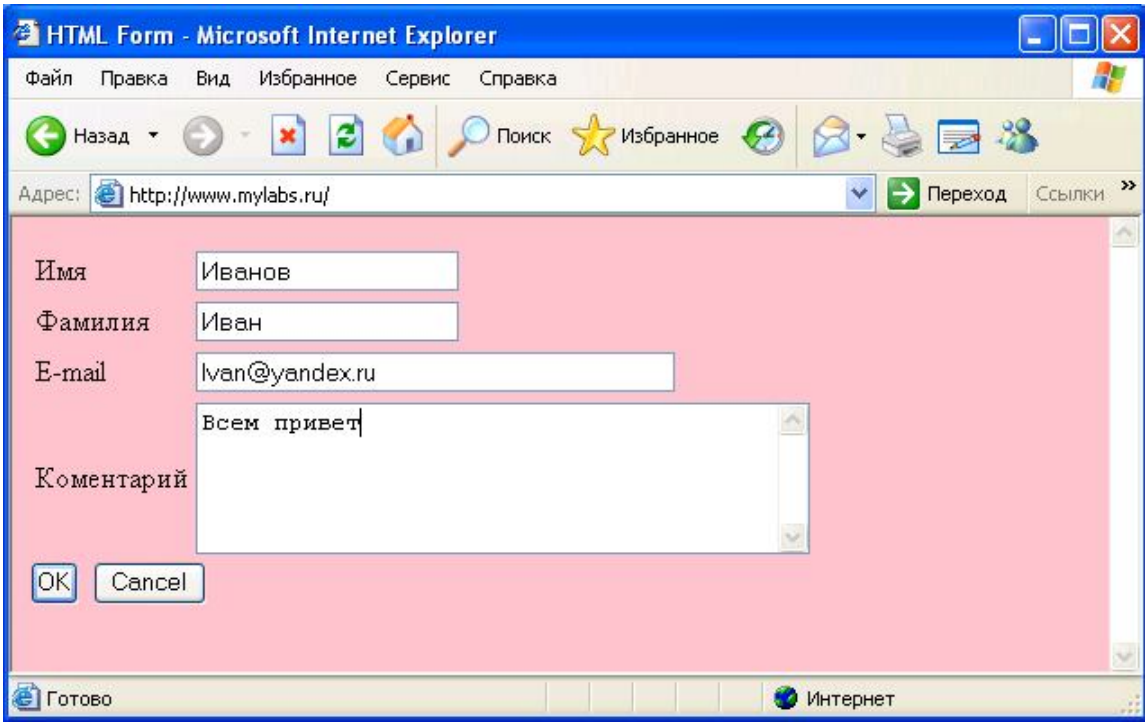
1. Рассмотрим пример создания формы (index.html), в которую пользователем вводятся данные:

Имя;

Фамилия;

Адреса электронной почты;

Текст комментария.

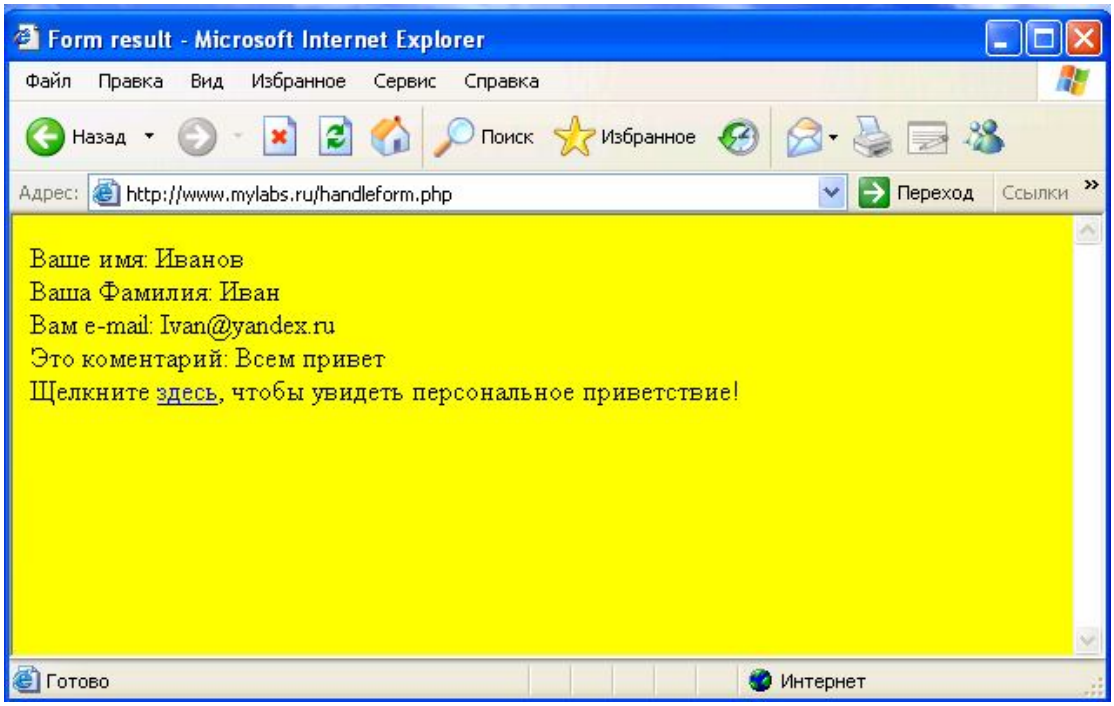


index.html

```
<html>
<head>
<title> HTML Form </title>
</head>
<body bgcolor=pink>
<form action="handleform.php" method=POST>
<table>
<tr>
<td> Имя</td>
<td><input type=text name="FirstName" size=20></td>
</tr>
<tr>
<td>Фамилия</td>
<td><input type=text name="LastName" size=20></td>
</tr>
<tr>
<td>E-mail</td>
<td><input type=text name="Email" size=40></td>
```

```
</tr>
<tr>
<td> Комментарий</td>
<td><textarea name="Comments" rows=5 cols=40></textarea></td>
</tr>
</table>
<input type=submit name="Submit" value="OK"> 
<input type=reset name="Reset" value="Cancel">
</form>
</body>
</html>
```

Затем PHP-сценарий (phandleform.php) получаетя данные с формы и отображает извлеченные из формы данные в окне браузера.



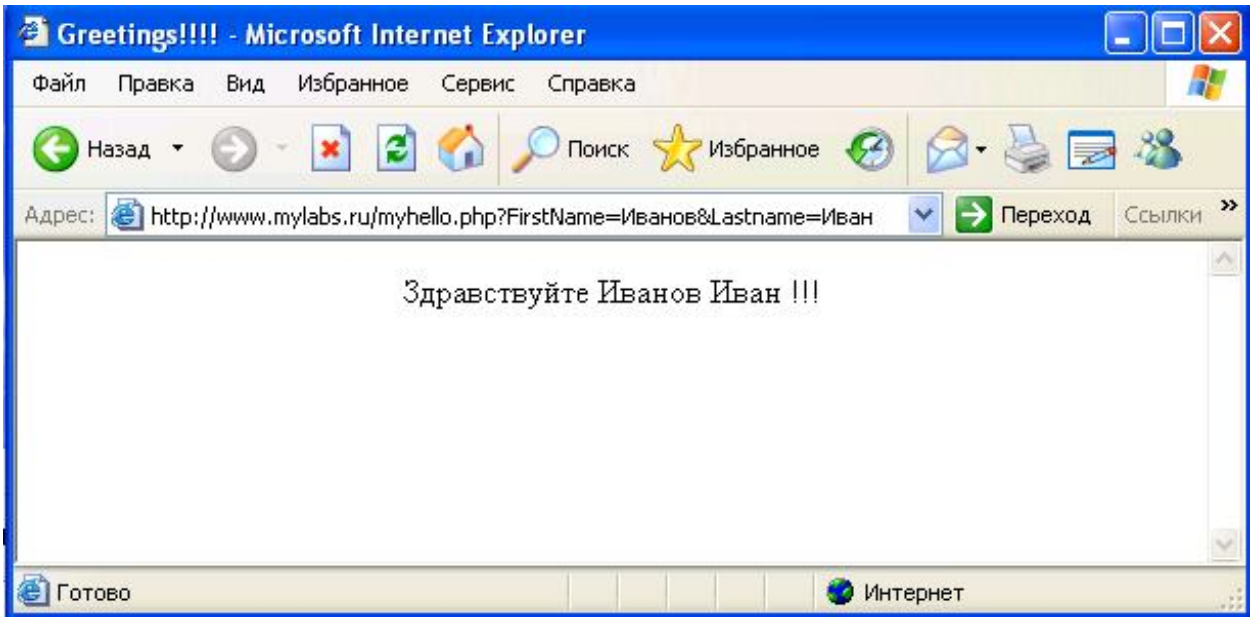
handleform.php
<html> <head> <title> Form result</title> </head> <body bgcolor=yellow> <?PHP print("Ваше имя: $_POST[FirstName] "); print("Ваша Фамилия: $_POST[LastName] "); print("Вам e-mail: $_POST[Email] "); print("Это комментарий: $_POST[Comments] "); print("Щелкните <A href=myhello.php?FirstName=$_POST[FirstName]&Lastname=$_POST[LastName]>здесь, чтобы увидеть персональное приветствие!"); ?> </body>

```
</html>
```

Далее сценарий при щелчке на ссылке, генерирует отправку на еще один PHP-файл (myhello.php), который отображает персональное приветствие пользователю.

myhello.php

```
<html>
<head>
<title>Greetings!!!!</title>
</head>
<body>
<?PHP
print("<center> Здравствуйте $_GET[FirstName] $_GET[Lastname] !!!</center>");
?>
</body>
</html>
```



Поменяйте в файле index.html значение атрибута method с POST на GET и посмотрите что изменилось.

2. Рассмотрим создание формы и PHP-сценария в одном файле

anketa.php

```
<head>
<title> Anketa</title>
</head>
<body bgcolor=yellow>
<?PHP
```

```
if (!isset($_POST[submit]))
{
print("<form action=\"anketa.php\" method=POST>");
print("<table> <tr><td> Имя</td>");
print("<td><input type=text name=\"FirstName\" size=20></td></tr>");
print("<tr><td>Фамилия</td>");
print("<td><input type=text name=\"LastName\" size=20></td></tr>");
print("<tr><td>E-mail</td>");
print("<td><input type=text name=\"Email\" size=40></td></tr>");
print("<tr><td> Комментарий</td>");
print("<td><textarea name=\"Comments\" rows=5
cols=40></textarea></td></tr></table>");
print("<input type=submit name=\"submit\" value=\"OK\"> ");
print("<input type=reset name=\"Reset\" value=\"Cancel\">");
print("</form>");
}
else
{
$FirstName=trim($_POST[FirstName]);
$Lastname=trim($_POST [LastName]);
$Email=trim($_POST [Email]);
$Comments=trim($_POST[Comments]);
$Name=$_POST[FirstName]." ".$_POST[LastName];
print("Ваше имя: $FirstName <br>");
print("Ваша фамилия: $Lastname <br>");
print("Ваш e-mail is: $Email <br>");
print("Комментарий: $Comments <br>");
print("Щелкните  <A href=\"myhello.php?Name=$Name\">здесь</a>, чтобы увидеть
персональное приветствие!");
}
?>
</body>
</html>
```

Отредактируйте файл myhello.php следующим образом

myhello.php

```
<html>
<head>
<title>Greetings!!!!</title>
</head>
<body>
<?PHP
print("<center> Здравствуйте $_GET[Name] !!!</center>");
?>
</body>
</html>
```

Наберите в браузере www.mylabs.ru/anketa.php и проверьте результат.

Примечание:

1. Функция trim –*принимает в качестве своего единственного аргумента строку, и удаляет из нее пробелы слева и справа.*
2. Функция isset –*определяет, установлена ли переменная.*

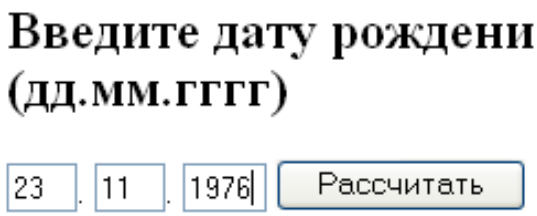
Синтаксис

bool isset ($var)

Возвращает TRUE, если var *существует, в противном случае возвращается FALSE.*

ЗАДАНИЕ

1. Создайте форму для ввода даты рождения и напишите скрипт для вычисления возраста и вывода его на экран (используйте метод GET).



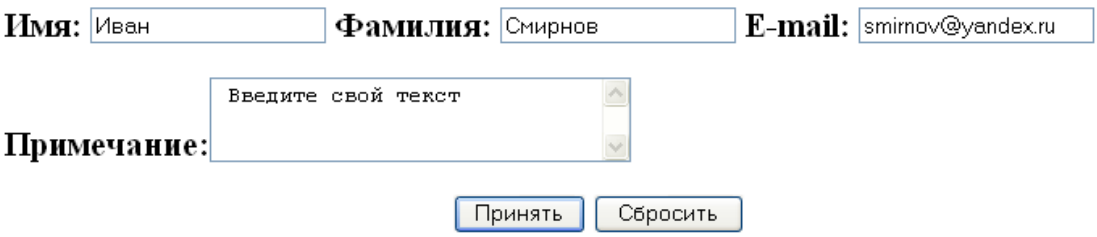
Примечание: *Для определения текущей даты используйте функцию* date(“m.d.Y”). *Чтобы выделить число, месяц и год допускается следующее использование:*

date(“d”) – *возвращает число текущего месяца;*

date(“m”) – *возвращает номер месяца текущего года;*

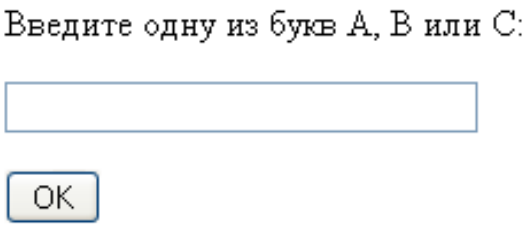
date(“Y”) – *возвращает текущий год.*

2. Разработайте форму как показано ниже



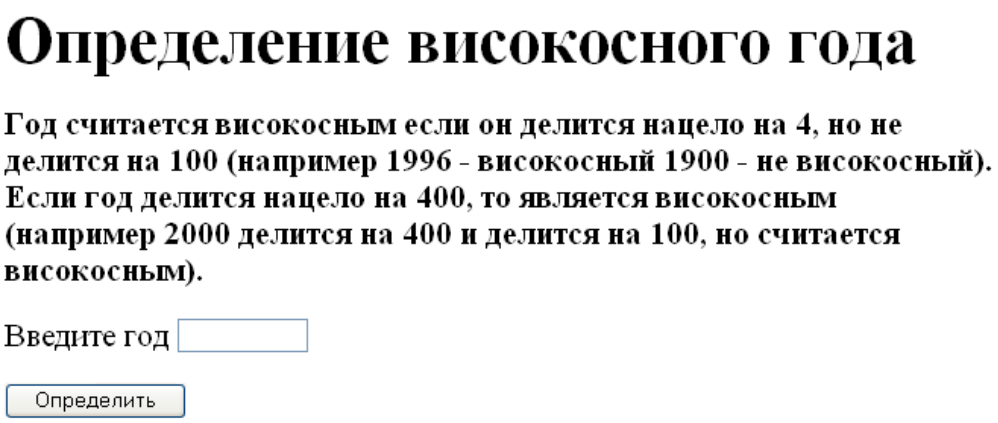
При нажатии на кнопку **Принять**, если поля **Имя** и **Фамилия** не пустые, то содержимое формы должно отображаться в новом окне в противном случае должно появиться окно о необходимости заполнить соответствующие поля (используйте метод POST).

3. Создайте форму

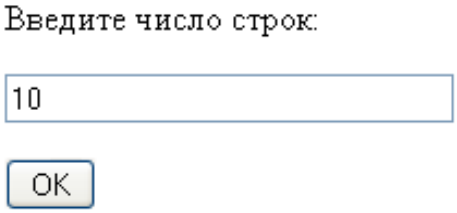


и напишите скрипт, который, в зависимости от введенной буквы выводит текст: "Вы ввели букву A" или "Вы ввели букву B" или "Вы ввели букву C".

4. Создайте форму как показано ниже и напишите скрипт определяющий по нажатию на кнопку **Определить** является ли введенный год високосным.



5. Создайте форму вида



и напишите скрипт выводящий числа в виде:

0
01
012
0123
01234
012345
0123456
01234567
012345678
0123456789