Титульный лист материалов по дисциплине

ДИСЦИПЛИНА **Базовые и прикладные информационные технологии**
(полное наименование дисциплины без сокращений)

ИНСТИТУТ **информационных технологий**

КАФЕДРА **практической и прикладной информатики**
полное наименование кафедры)

ВИД УЧЕБНОГО МАТЕРИАЛА **Практические занятия**
(в соответствии с пп.1-11)

ПРЕПОДАВАТЕЛЬ **Борзых Никита Юрьевич**
(фамилия, имя, отчество)

СЕМЕСТР **1, 2024-2025**
(указать семестр обучения, учебный год)

Практическая работа №1.

Часть 1. Знакомство с языком PHP. Управляющие конструкции.

Цель работы: Приобретение навыков программирования линейных, ветвящихся и циклических алгоритмов.

Задачи: Изучить базовые элементы языка:

1. Типы данных, идентификаторы, комментарии, объявление переменных (область видимости).

2. Управляющие конструкции языка. Константы. Циклы.

Теоретический материал:

1. Изучить типы данных (скалярные, смешанные, специальные).

Язык PHP предоставляет 4 скалярных типа данных:

- boolean - логический, принимает значения true (истина) и false (ложь);
- integer - целочисленный, обычно 4-байтовый, но размерность зависит от платформы;
- float - число с плавающей точкой, обычно представлено с двойной точностью, соответствующей типу double в Си-подобных языках;
- string - строка, на большинстве платформ размер ограничен 2 гигабайтами, по умолчанию символу соответствует 1 байт, т.е. в текущих версиях PHP нет встроенной поддержки Юникода.

Также определены два смешанных типа:

- array - массив пар "ключ" - "значение", если ключи не указаны, соответствует массиву в языке C++ (нумерация элементов с нуля);
- object - объект, используется в объектно-ориентированном программировании с помощью классов.

Два специальных типа, resource и null служат, соответственно, для создания ссылок на внешние ресурсы и представления переменной без значения:

Условные операторы: Оператор if. Структуру оператора if можно представить следующим образом:

if (выражение) блок_выполнения. Структуру оператора if, расширенного с помощью оператора else, можно представить следующим образом:

if (выражение) блок_выполнения else блок_выполнения1. Оператор elseif. Еще один способ расширения условного оператора if – использование оператора elseif. Elseif – это комбинация else и if. Как и else, он расширяет if для выполнения различных действий в том случае, если условие, проверяемое в if, неверно. Но в отличие от else, альтернативные действия будут выполнены, только если elseif-условие является верным:

if (выражение) блок_выполнения
elseif(выражение1) блок_выполнения1
...
else блок_выполненияN.

Оператор switch:

switch (выражение или переменная) {
case значение1:
блок_действий1
break;
case значение2:
блок_действий2
break;
...
default:
блок_действий_по_умолчанию
}

Циклы.

В PHP существует несколько конструкций, позволяющих выполнять повторяющиеся действия в зависимости от условия. Это циклы while, do..while, foreach и for.

while
Структура:
while (выражение) { блок_выполнения }
либо
while (выражение): блок_выполнения endwhile;

Циклы do..while очень похожи на циклы while, с той лишь разницей, что истинность выражения проверяется в конце цикла, а не в начале.

```
do {блок_выполнения} while (выражение);
<?
// эта программа напечатает число 12, несмотря на то
// что условие цикла не выполнено
$i = 12;
do{
if ($i % 2 == 0) print $i;
// если число четное, то печатаем его
$i++;
// увеличиваем число на единицу
}while ($i<10)
?>
```

for: Структура:

for (выражение1; выражение2; выражение3) {блок_выполнения} либо

for (выражение1; выражение2; выражение3): блок_выполнения endfor;

foreach: Еще одна полезная конструкция. Предназначена исключительно для работы с массивами. Синтаксис:

foreach ($array as $value) {блок_выполнения}

либо

foreach ($array as $key => $value)

{блок_выполнения}

Операторы передачи управления: break и continue. Операторы включения include. Оператор include позволяет включать код, содержащийся в указанном файле, и выполнять его столько раз, сколько программа встречает этот оператор. Включение может производиться любым из перечисленных способов:

```
include 'имя_файла';
include $file_name;
include ("имя_файла");
```

Основные математические функции:

Функция	Описание
abs( )	Абсолютное значение числа.
acos( )	Арккосинус, выраженный в радианах.
asin( )	Арксинус, выраженный в радианах.
atan( )	Арктангенс, выраженный в радианах.
atan2( )	Арктангенс для координат x и y, выраженный в радианах. Отличие от выражения **atan(y/x)** состоит в том, что знаки обоих параметров используются для определения квадранта результата.
base_convert( )	Переводит число из одной системы счисления в другую. ргументы: переводимое число, система счисления, из которой переводят, система счисления, в которую переводят.
bindec( )	Десятичный эквивалент двоичной строки. Наибольшее конвертируемое число содержит 31 разряд, что соответствует 2147483647.
ceil( )	Округление числа в большую сторону.
cos( )	Косинус аргумента, выраженного в радианах.

Функция	Описание
decbin( )	Двоичный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647, или 31 разряд.
dechex( )	Шестнадцаричный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647 или 7fffffff в шестнадцаричном выражении.
decoct( )	Восьмеричный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647 или 17777777777 в восьмеричном выражении.
deg2rad( )	Преобразует градусы в радианы.
exp( )	Экспонента числа.
floor( )	Округление числа в меньшую сторону.
getrandmax( )	Максимальное число, которое может быть получено в результате вызова функции **rand( )**.

Функция	Описание
	десятичной точки (необязательно), символ разграничения тысяч (необязательно).
octdec( )	Десятичный эквивалент восьмеричного числа, представленного строкой. Наибольшее конвертируемое число составляет 17777777777 или 2147483647 в десятичном выражении.
pi( )	Приближенное значение числа π.
pow( )	Возведение в степень. Аргументы: основание и показатель степени.
rad2deg( )	Преобразует радианы в градусы.
rand( )	Псевдослучайное число. Необязательные аргументы указывают диапазон допустимых значений. Если их не задавать, то число выбирается из диапазона от 0 до RAND_MAX. Перед использованием этой функции необходимо установить начальное число с помощью функции **srand( )**.
round( )	Округление числа до ближайшего целого.
sin( )	Синус аргумента, выраженного в радианах.
sqrt( )	Квадратный корень числа.

Порядок выполнения работы:

- Ознакомление с содержимым теоретической части.

Задание 1. Определить переменные всех типов с проверкой вывода заданных значений и возможностью переопределения. Реализовать присваивание по ссылке, т.е. запись в одну переменную адрес другой.

Примеры выполнения:

1)

```
$a = 5; //целочисленная величина
$b = 'String'; /* строка */
$c = 3.14; #вещественное число
echo "a=$a,b=$b,c=$c";
```

2) Присваивание по ссылке, позволяющее записать в одну переменную адрес другой:

```
$var = 'test';
$var2 = &$var;
```

3) "переменная переменных", позволяет использовать значение одной переменной в качестве имени другой и создающее дополнительные возможности косвенной адресации данных:

```
$var=1;
$list = 'var';
$$list = 2;
echo "$var"; //2
```

Таким образом, конструкция $$Имя_переменной означает "взять переменную, имя которой получено из переменной $Имя_переменной".

4) для описания констант служит конструкция define, в которой нужно указать имя и значение константы:

```
define('login','admin');
define ("password","12345");
echo "<br>" .login;
echo "<br>" .password;
```

Задание 2. Создать php-скрипт по реализации математических операций и условного оператора.

1) в php определен оператор присваивания (=) и 5 бинарных арифметических операций - умножение (*), деление (/), сложение (+), вычитание (-), взятие остатка от деления (%). В отличие от C++, деление целых чисел не дает целого результата:

```
<?php
 $a= 3/5;
 echo "$a"; //0.6
?>
```

Как и в C++, разрешены префиксный и суффиксный инкремент и декремент, присваивание "цепочкой" и "на лету":

```
$c=$d=0;
  echo "$c,$d "; //0,0
  $d=8+($c=3);
  echo "$c,$d"; //3,11
```

Результаты операций сравнения интерпретируются и как значения 1/0, и как true/false:

```
<?php
 $a=$b=0;
 echo $a==$b; //1
 if (($a==$b)==true) echo "1"; //1
?>
```

2) унарный оператор подавления ошибки @ может использоваться перед любым выражением, имеющим значение. Например, код:

```
<?php
 $x=0; $a=7;
 @($a/=$x);
 echo "$a"; //$a не будет выведено
?>
```

Задание 3. Реализовать скрипты php с использованием циклов: while, do..while, foreach и for.

1)

```
$k=0;
 for ($i=1; $i<=10; $i++) {
  echo '<br>$i='.$i;
  for ($j=1; $j<=10; $j++) {
   echo ' <br>$j='.$j;
   $k++;
   if ($k==10) break 2;      //выход сразу из 2 циклов!
```

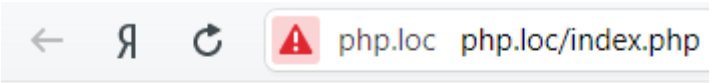
```
  }
 }
```
2)
```
$i = 1;
while ($i < 10) {
if ($i % 2 == 0) print $i;
// печатаем цифру, если она четная
$i++;
```
3) Основной формой альтернативного синтаксиса является изменение открывающей фигурной скобки на двоеточие (:), а закрывающей скобки на конструкции endif;, endwhile;, endfor;, endforeach; или endswitch; соответственно.
```
for ($i = 1; $i <= 5; $i++) :
  echo 'куб числа '.$i.' = '.pow($i,3).'<br>';
 endfor;
```
4) foreach – предназначен исключительно для работы с массивами.
```
$names = array("Александра","Петр","Игорь");
foreach ($names as $val) {
echo "Здравствуйте, $val <br>";
// выведет всем приветствие
}
foreach ($names as $k => $val) {
// кроме приветствия,
// выведем номера в списке, т.е. ключи
echo "Здравствуйте, $val ! В списке Вы под номером $k <br>";

}
```
5)
```
for ($i=100;$i<=999;$i++)
{
```

```
    $a=$i/100; $b=$i/10%10; $c=$i%10;
    if ($i==100*$c+10*$b+$c)
    echo 'Число '.$i .'удовлетворяет условию';
}
```

Задание 4: Вывести трехзначные четные числа, которые делятся без остатка на заданное произвольное число:

```
<?php
$n=rand(10,99);
echo 'Произвольное число: '.$n;
echo'<br> Числа, удовлетворяющие условию: <br>';
for($i=100;$i<=999;$i++){
    if ($i%$n==0 && $i%2==0){
     echo $i.'<br>';
    }
  }
  ?>
```



Произвольное число: 83
Числа удовлетворяющие условию:
166
332
498
664
830
996

Задания 1-4 в отчет не включаются, по ним при защите работы преподавателем выдаются дополнительные задания или вопросы.

- Выполнение заданий № 5- 7 в соответствии с индивидуальным вариантом:

Задание 5. Программирование линейных алгоритмов.

Задание 6. Программирование ветвящихся алгоритмов.

Задание 7. Программирование циклических алгоритмов.

Распределение заданий по вариантам:

№ вар.	Задание 5	Задание 6	Задание 7
1	Вычислить длину окружности и площадь круга одного и того же заданного радиуса.	Определить, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.	Дано действительное x. Вычислить: $y(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} - \frac{x^9}{9!}$.
2	Найти площадь равностороннего треугольника, его высоты, радиусы вписанной и описанной окружностей по заданной стороне.	Даны три положительных числа. Проверить, могут ли они быть длинами сторон треугольника. Если да, то тип треугольника (остроугольный, прямоугольный, тупоугольный).	Дано действительное n. Вычислить: $\sum_{n=1}^{\infty} \frac{5}{25n^2 - 5n - 6}$
3	Вычислить расстояние между двумя точками в пространстве с заданными координатами.	Даны две точки с заданными: абсциссой и ординатой. Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.	Дано натуральное число n. Переставить местами первую и последнюю цифры этого числа.
4	Найти периметр и площадь произвольного треугольника по известным координатам вершин (абсциссы и ординаты).	Подсчитать количество целых чисел среди заданных трех чисел.	Даны действительное число a, натуральное число n. Вычислить: $P = a\,(a-n)\,(a-2n)\times\ldots\times(a-n2)$
5	Найти площадь полной поверхности и объем куба по известному ребру.	Даны три положительных числа. Проверить, могут ли они быть длинами сторон треугольника. Если да, то вычислить радиус вписанной окружности.	Дано натуральное число n. Переставить его цифры так, чтобы образовалось максимальное число, записанное теми же цифрами.
6	Найти сумму членов арифметической прогрессии, если известны ее первый член, знаменатель и число членов прогрессии.	Даны три положительных числа. Проверить, могут ли они быть длинами сторон треугольника. Если да, то вычислить площадь этого треугольника.	Найти все двузначные числа, сумма квадратов цифр которых кратна M .
7	Найти периметр и радиус вписанной окружности прямоугольного треугольника по заданным длинам двух катетов.	Подсчитать количество целых отрицательных чисел среди заданных трех чисел.	Дано действительное x. Вычислить: $\frac{(x-1)(x-3)(x-7)\times\ldots\times(x-63)}{(x-2)(x-4)(x-8)\times\ldots\times(x-64)}$
8	Найти площадь и радиус описанной окружности произвольного треугольника если известны две стороны и угол между ними.	По заданному радиусу круга и стороне равностороннего треугольника определить, поместится ли правильный треугольник в этом круге.	Составить программу, которая печатает таблицу квадратов от 10 до 30.
9	Вычислить площадь боковой поверхности и объем цилиндра с заданными радиусом и высотой.	Подсчитать количество целых положительных чисел среди заданных трех чисел.	Найти сумму всех n-значных чисел $(1 \le n \le 4)$

10	Найти площадь боковой поверхности и объем конуса по заданным радиусом и высотой.	Даны три положительных числа. Проверить, могут ли они быть длинами сторон треугольника. Если да, то вычислить радиус описанной окружности.	Составить программу, которая печатает таблицу умножения.

Содержание отчета:

1. Титульный лист.

2. Цель работы, задание.

3. Исходный текст программы на PHP **для заданий 5-7**.

4. Скриншот сгенерированной страницы.

5. Выводы.

Контрольные вопросы:

1. Основные конструкции.

2.Типы данных. Идентификаторы.

3. Объявление переменных, область видимости.

4. Глобальные переменные.

5.Переключение и преобразование типов.

6. Присваивание по значению и по ссылке.

7.Стандартные переменные. Константы.

8. Операторы: математические, строковые, логические, поразрядные, операторы присваивания, инкремента и декремента, сравнения.

9. Управляющие конструкции.

10. Проверка условий. Альтернативное ограничение блоков.

11. Циклы while, do..while, for, foreach.

12. Операторы switch, break, continue.

Практическая работа №1.

Часть 2. Обработка запросов с помощью PHP.

Цель работы: Приобретение навыков создания программ для организации обмена информацией между WEB-сервером и клиентом.

Задачи:

1. Изучить конструкции HTML для организации обмена между сервером и клиентом.
2. Разобрать методы передачи данных GET и POST.
3. Изучить возможности PHP для получения информации от пользователя.

Теоретический материал

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). Сообщения по сети при использовании протокола HTTP передаются в формате, схожем с форматом почтового сообщения Internet (RFC-822) или с форматом сообщений MIME (Multipurpose Internet Mail Exchange). HTTP используется для коммуникаций между различными пользовательскими программами и программами-шлюзами, предоставляющими доступ к существующим Internet- протоколам, таким как SMTP (протокол электронной почты), NNTP (протокол передачи новостей), FTP (протокол передачи

файлов), Gopher и WAIS. HTTP разработан для того, чтобы позволять таким шлюзам через промежуточные программы-серверы (proxy) передавать данные без потерь. Протокол реализует принцип запрос/ответ. Запрашивающая программа – клиент инициирует взаимодействие с отвечающей программой – сервером и посылает запрос, содержащий:

- *метод доступа;*
- *адрес URI;*
- *версию протокола;*
- *сообщение (похожее по форме на MIME) 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 форме, похожей на MIME), в которое входит информация сервера, метаинформация (т.е. информация о содержании сообщения) и тело сообщения.*

В протоколе не указывается, кто должен открывать и закрывать соединение между клиентом и сервером. На практике соединение, как правило, открывает клиент, а сервер после отправки ответа инициирует его разрыв.

Простой запрос содержит метод доступа и адрес ресурса:

<Простой-Запрос> := <Метод> <символ пробел>

<Запрашиваемый-URI> <символ новой строки>

В качестве метода могут быть указаны GET, POST, HEAD, PUT, DELETE и другие. Полный запрос содержит строку состояния, несколько заголовков (заголовок запроса, общий заголовок или заголовок содержания) и, возможно, тело запроса.

<Полный запрос> := <Строка Состояния>

(<Общий заголовок>|<Заголовок запроса>|

<Заголовок содержания>)

<символ новой строки>

[<содержание запроса>]

Формы — это совокупность стандартных HTML-конструкций ввода текстовой и прочей информации и программы-обработчика этой информации, работающей на Web-сервере. Иными словами, пользовательская форма (или HTML-форма) служит для передачи информационных данных серверу.

Результат конструкций языка разметки HTML интерпретируется браузером, с помощью которого пользователь электронного документа получает информацию. Таким образом, объединив все эти формулировки, можно сказать, что HTML-форма выступает в роли посредника между пользователем и сервером. Посетитель Web-страницы вводит в HTML-форму определенные данные, которые обрабатываются программой и отсылаются на Web-сервер. Все эти действия укладываются в три стадии:

1. Ввод пользователем информации.
2. Обработка введенной информации программой, установленной на сервере.
3. Получение результата отправления введенной информации на Web-сервер (открытие нового HTML-документа, переадресация на предыдущую страницу и пр.).

В качестве программы-обработчика чаще всего выступает CGI-сценарий (скрипт, который обычно разрабатывается на языке Perl или C/C++ и который взаимодействует со специальным компонентом Web-сервера —Common Gateway Interface) или программы, написанные на основе таких серверных языков программирования, как PHP, ASP, JSP и др.

Пример формы ввода логина и пароля на сайте:

```
<form action="obrabotchik1.php" method="post"><div>
Логин: <input type="text" value="" name="login"/>
Пароль: <input type="password" value="" name="password"/>
<input type="submit" value="Проверить" name="button"/>
</div></form>
```

Пример обработки формы:

```
<? $userLogin = $_POST["login"];
 $userPassword = $_POST["password"];
 if ($userLogin=="Vasiliy" && $userPassword=="12345")
echo "Здравствуйте, Василий! Логин и Пароль верны.";
else echo "Ошибка в вводе Логина или Пароля. Василий вы забыли пароль?";
?>
```

Поле **text -** элемент текстовой строки:

```
<input type=text name=имя_параметра [value=значение] [size=размер_поля]
[maxlen=длина_поля]>
```

radio -списки-переключатели:

```
 <tr>
   <td>Ваш пол:</td>
</tr>
  <tr> <td>
    <input type="RADIO" CHECKED name="pol" value="Мужской"> Мужской </td> </tr>
   <tr> <td><input type="RADIO" name="pol" value="Женский"> Женский </td> </tr>
 <tr> <td><br> <br>
```

select - выпадающий список:

```
 <tr> <td>Образование :<select name="edu" size="1">
 <option selected value="Высшее">Высшее</option>
 <option value="Незаконченное высшее">Незаконченное высшее</option>
<option value="Среднее полное">Среднее полное</option>
 <option value="Среднее неполное">Среднее неполное</option>
 </select><br>
```

checkbox - списки вариантов:

```
 <tr> <td><input type="CHECKBOX" name="computer" checked value="ON">
Наличие компьютера </td> </tr>
```

submit - кнопка отправки формы:

```
<input type=submit [name=go] value=Отправить>/
```

reset - кнопка сброса формы:

<input type=reset value=Сброс>

image - кнопка с изображением:

<input type=image name=имя src=рисунок>

password - поле пароля:

<INPUT TYPE="password" SIZE="30" NAME="password">

hidden - скрытое текстовое поле:

<input type=hidden name=имя_параметра value=значение>

Фильтрация пользовательских данных, полученных методом GET или POST.

Обычно требуются следующие этапы обработки:

1. Удаление лишних разделителей в пользовательском вводе (пробелов, переводов строки). Реализуется стандартной функцией trim и с помощью регулярных выражений.

2. Удаление из ввода пользовательской разметки HTML-тегами (функция strip_tags) или превращение ее в печатаемые символы HTML (функция htmlspecialchars; при ее применении, например, символ "<", прочитанный из поля ввода формы, превратится в HTML-сущность "<" и не сможет изменить разметку создаваемого скриптом документа).

3. В PHP версий ниже 5.4.0 есть несколько настроек, отвечающих за отображение апострофов и кавычек (традиционное название - "магические кавычки"). При неудачном стечении обстоятельств, например, двойная кавычка из пользовательского ввода может "экранироваться" и превратиться в \", испортив внешний вид текста. Стандартной функции, решающей все проблемы с кавычками, нет, напишем для этой цели собственную функцию с именем magic.

4. Если это уместно, над полученными из массива $_GET или $_POST элементами формы следует выполнить явное преобразование типов данных, например, для целого значения - функцией intval, для строки, передаваемой в SQL-запрос к базе данных - функцией mysql_real_escape_string и т.д.

5. В профессиональных скриптах чаще всего необходимо проверить URL-адрес, с которого была отправлена форма - ведь ничто не мешает злоумышленнику написать скрипт, который будет отправлять данные нашему скрипту программно, перегружая таким образом сервер или стремясь подобрать пароль пользователя. Простейшие проверки такого рода включают в себя контроль значения $_SERVER['HTTP_REFERER'] и/или IP-адреса пользователя. К сожалению, HTTP - открытый протокол, и подделать можно все поля его заголовка, включая адрес страницы, с которой отправлена форма и IP-адрес отправителя.

Поэтому достаточно надежной защитой от программной атаки на HTML-формы в настоящее время можно считать лишь отправку вместе с данными формы некоторой "капчи" (captcha) - задачи, которую с легкостью может решить человек, но со значительно меньшей вероятностью может решить программа (например, распознавание текста, выведенного поверх изображения).

Порядок выполнения работы:

- Ознакомление с содержимым теоретической части.

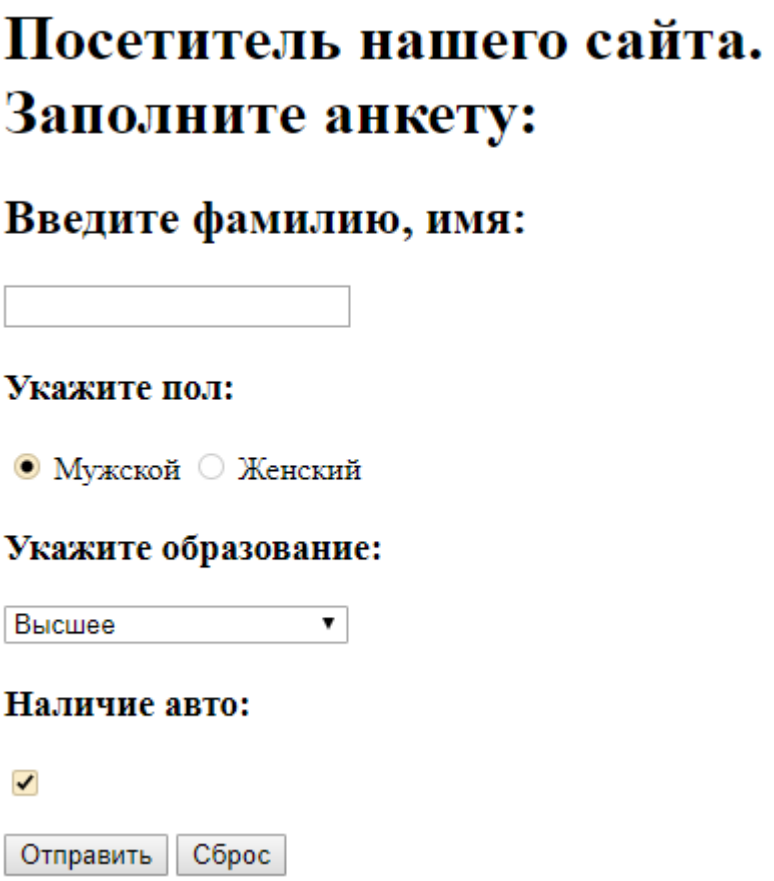
Задания 1-2 в отчет не включаются, по ним при защите работы преподавателем выдаются дополнительные задания или вопросы.

Задание 1. Создать HTML-документ, содержащей форму: Заполнение анкеты.

```
<FORM action="index2.php" method="post">
   <h1> Заполните анкету:<br></h1>
   <h2> Введите фамилию, имя:<br></h2>
   <input type='text' name='fio' ><br>
   <h3> Укажите пол:<br></h3>
   <input type="RADIO" CHECKED name="pol" value="Мужской"> Мужской
   <input type="RADIO"  name="pol" value="Женский"> Женский
```

```
 <h3>  Укажите образование:<br></h3>
<select name="edu" size="1">
   <option selected value="Высшее">Высшее</option>
   <option value="Незаконченное высшее">Незаконченное высшее</option>
   <option value="Среднее полное">Среднее полное</option></select>
  <h3>  Наличие авто:<br></h3>
  <input type="CHECKBOX" name="avto" checked value="ON">

  <div align="left"><left><p>
        <input type="submit" value="Отправить">
        <input type="reset" value="Сброс"> <br> </p> </left></div>
</FORM>
```



Задание 2. Разработать PHP-скрипт, анализирующий введенные данные и генерирующий страничку с анализом введенных данных пользователем:

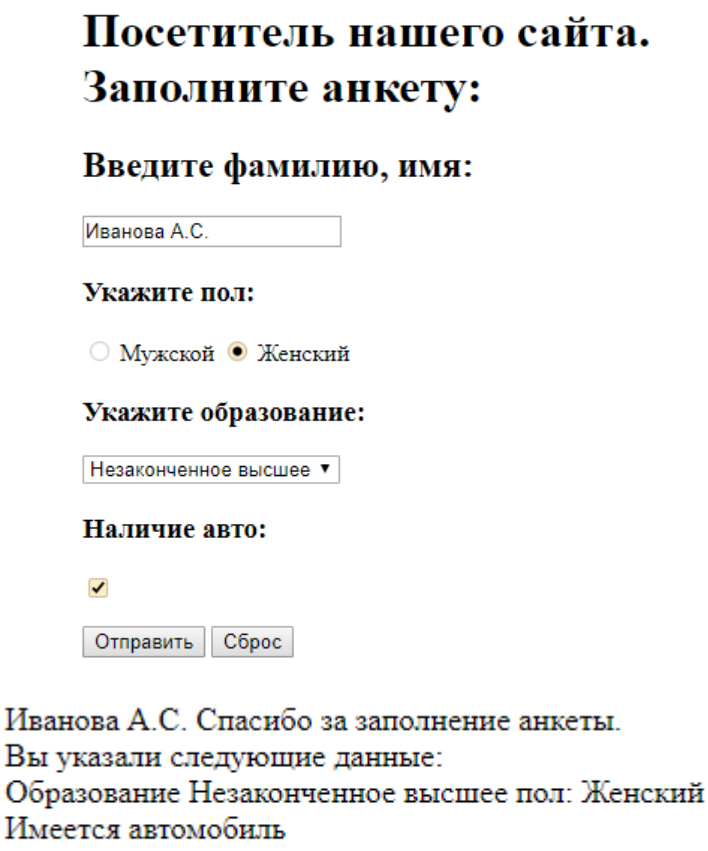
```
<?php
```

```
$k=$_POST["fio"];
$zn=$_POST["pol"];
$n=$_POST["edu"];
$f=$_POST["avto"];
echo $k."  Спасибо за заполнение анкеты. <br>";
echo 'Вы указали следующие данные:<br>';
echo 'Образование '.$n.'  пол: '.$zn.'<br>';
if ($f=="ON")
{
   echo 'Имеется автомобиль<br>';
}
```

Или:

```
<?php

echo $_POST["fio"]."  Спасибо за заполнение анкеты. <br>";
echo 'Вы указали следующие данные:<br>';
echo 'Образование '.$_POST["edu"].'  пол: '.$_POST["pol"].'<br>';
if ($_POST["avto"]=="ON")
{
   echo 'Имеется автомобиль<br>';
}
```



Задание 3: Разработать приложение, в котором:

1. Создается форма для введения пользователем данных:

2. PHP-сценарий получает данные с формы.

3. Отображает извлеченные из формы данные в окне браузера.

4. Сценарий генерирует отправку на еще один PHP-файл, который отображает персональное приветствие пользователю.

Внутри формы могут располагаться следующие элементы интерфейса:

поля ввода;

скрытые поля ввода;

кнопки;

переключатели;

флажки;

выпадающие списки.

Задания по вариантам:

1. Форма записи на курсы повышения квалификации.

2. Форма регистрации участника конференции.

3. Форма отзыва о предоставленных услугах.

4. Анкета по завершении прохождения курсов.

5. Форма заказа товара.

6. Регистрация на сайте.

7. Регистрация на сайте с указанием обязательной и дополнительной информации.

8. Заполнение анкеты при приеме на работу.

9. Форма записи на прием в частную клинику.

10. Форма записи в страховую компанию.

Задание 4: Решить задания первой части практической работы, используя формы.

Содержание отчета:

1. Титульный лист.

2. Цель работы, задание.

3. Исходный текст программы на PHP **для заданий 3-4.**

4. Скриншот сгенерированной страницы.

5. Выводы.

Контрольные вопросы:

1. Использование HTML-форм для передачи данных на сервер.

2. Методы передачи данных.

3. Основные элементы формы.

4. Обработка запросов с помощью PHP.