# Основы языка PHP – управляющие конструкции и функции

✔Управляющие конструкции – циклы
- for
- while
- do…while
- foreach

✔Функции:
- описание функций
- область видимых переменных

# Введение в PHP

## Циклы

Циклы предназначены для многократного исполнения набора инструкций.

**Цикл for**

В цикле for указывается начальное и конечное значения счетчика, а так же шаг, с которым счетчик будет изменяться. Изменяться счетчик может как в положительную, так и отрицательную сторону. Действия выполнятся столько раз, сколько итераций пройдет от начального значения счетчика до достижения конечного, с указынным шагом.

```
for(начало;конец;шаг){
Действие;
    ...
}
```

```
for ($i = 1; $i <= 5; $i++){
$sum += $i;
echo $sum;
}
```

# Введение в PHP

## Циклы

**Цикл while**

Действия будут выполняться до тех пор, пока условие истинно.

Цикл while является циклом с предусловием.

```
while (условие) {
    Действие;
    ...

}
```

```
while ($state == 'Солнце высоко') {
    echo 'Рабочий день продолжается';
    $state = 'Солнце заходит';

}
```

# Цикл do...while

Цикл do…while является циклом с постусловием. Это значит, что сначала будет выполняться действие, а потом проверяться условие.

Таким образом действие всегда выполнится минимум один раз.

```
do{
    Действие;
    ...
} while (условие);
```

```
do{
    echo 'Пиф-паф';
} while ($state =='Живой');
```

# Введение в PHP

## Управление циклами

Break прерывает работу цикла. Интерпретатор перейдет к выполнению инструкций, следующих за циклом.

```
$index = 1;
while ($index < 10){
    echo "$index <br>";
    $index++;
    if ($index == 5)
        break;
}
```

Continue прерывает выполнение текущей итерации цикла. Цикл продолжит выполняться со следующей итерации.

```
$index = 0;

while ($index < 10){
    $index++;
    if ($index == 5)
        continue;
        echo "$index
<br>";
}
```

# Введение в PHP

## Цикл foreach

Очень удобен при работе с массивами. Указанные действия выполняются для **каждого** элемента массива $array, при этом $key — номер элемента массива $array, $value — значение этого элемента.

```
foreach ($array as [ $key => ] $value){
    Действия;
    ...
}
```

```
<?php
    $pets[] = 'Собака';
    $pets[] = 'Кошак';
    $pets[] = 'Рыбка';
    foreach ($pets as $index => $value) {
        echo "Элемент №$index имеет значение:
\"$value\"<br>";
    }
?>
```

# Функции: описание и вызов

Функция – программных блок, который может многократно выполняться в любом месте сценария.

```
<?php
    function name([argument1][,argument2][,...]){
        // Тело функции
    }
?>
    /*Описание*/
    function printText(){
        echo “Hello, world!”;
    }
  /*Вызов функции*/
  printText();
  if(function_exists(“printText”)){} // Проверка
```

# Функции и их аргументы: описание и вызов

```
function printText($name){
    echo “Hello, $name!”;
}
// Вызываем функцию, вариант 1
print  (“Иван”);
// Вызываем функцию, вариант 2
$name = “Петр”;
printText($name);
// Вызываем функцию, вариант 3
$func = “printText”;
$func(“Игорь”);
```

# Функции: аргументы по умолчанию

```
function printText($name=“Гость”){
    echo “Hello, $name!”, “<hr>”;
}
    printText(“Иван”);
    printText(“Петр”);

    printText();// Hello, Гость!
```

# Функции и Code Reuse

## Области видимости

```php
<?php
    $a = 1;                 // Глобальная область видимости
    function Test(){
        echo $a;            // Локальная область видимости
    }
    Test();                 // Не выведет ничего
?>
```

Получить доступ к глобальным переменным из локальной области видимости можно следующими способами:

```php
<?php
    $a = 1; $b = 2;
    function Sum () {
        global $a, $b;
        return $b += $a;
    }
    echo Sum();
?>
```

```php
<?php   // Рекомендуемый вариант
    $a = 1; $b = 2;
    function Sum () {
        return $GLOBALS['b'] += $GLOBALS['a'];
    }
    echo Sum();
?>
```

# Функции и Code Reuse

## Статические переменные

Есть другой способ определить переменную так, чтобы ее значение хранилось вне функции, но было доступно также и внутри нее. Способ заключается в использовании статических переменных.

```
<?php
    function Test() {
        static $a = 0;
        echo $a;
        $a++;
    }
    Test();         // 0
    echo $a;        // Ошибка:$a нет в глобальной области
видимости
    Test();         // 1
    Test();         // 2; Переменная $a сохранила свое значение
?>
```

Статическая переменная как коробочка: в локальной области видимости она открыта, в глобальной - закрыта, но никуда не делась.

# Функции: возврат значений

```
function get_sum($number1, $number2){
        return $number1 + $number2;
}
$result = get_sum(10, 435);
echo $result;

// или
echo get_sum(10,435);
```

## Функции: передача аргументов по ссылке

```
function Test_1($a){
        $a++;
}
function test_2($a){
        $a++;
}
$i=0;
Test_1($i); // 1
echo  $i;
Test_2($i); //1
echo  $i;
Test_2(5); // Ошибка!!!
```

# Рекурсивный вызов функций

```
function factorial($n){
    if($n==0)   return 1;
        return $n*o factorial($n-1);
}

$result = factorial(5);
echo “5!=”.$result;
```

# Введение в PHP

## Практическая работа

1. Используя условный переход, выведите сообщение «Счастливчик!» если $age попадает в диапазон между 18 и 35. Если значение иное, выведите «Не повезло». Расширьте предыдущую конструкцию сообщением «Слишком молод», если $age в диапазоне между 1 и 17.

2. Используя циклы, сформируйте массив четных чисел из диапазона от 1 до 100. Выводя массив на экран, исключите из вывода все числа, которые не делятся на 5.

3. Создайте массив со следующими элементами: Name, Address, Phone, Mail и заполните его. С помощью цикла foreach осуществите форматированный вывод массива в виде: «элемент: значение».