

# **МДК 02.02. Web-программирование. Язык PHP**

**Работа со строками**

**Строка в PHP – это набор символов любой длины.**

### **Способы определения строк:**

1. в одинарных кавычках;

#### **Пример**

```
echo 'это простая строка';
```

```
$a = 'это простая строка';
```

2. в двойных кавычках;

#### **Пример**

```
$juice = "apple";
```

```
echo "He drank some $juice juice";
```

### 3. heredoc-синтаксис;

Строка в формате heredoc начинается с последовательности <<< и идентификатора (учитывается регистр символов). Затем записывается строка, а потом этот же идентификатор, закрывающий вставку. Строка с закрывающим идентификатором не содержит других символов, за исключением точки с запятой (;).

#### Пример

```
$str = <<<ABC
```

Пример строки,

охватывающей несколько строк,

с использованием heredoc-синтаксиса.

```
ABC;
```

## 4. nowdoc-синтаксис

Внутри синтаксиса Nowdoc не осуществляется подстановок.

Идентификатор заключается в одинарные кавычки.

### Пример

```
<?php  
echo $a=<<<'EOD'
```

Пример строки с использованием nowdoc-синтаксис

```
EOD;
```

```
?>
```

Если строка указывается в двойных кавычках, либо при помощи heredoc, переменные внутри нее обрабатываются.

## **Интерполяция**

**Интерполяция** – это замена переменной в строке ее содержимым.

Интерполяция является свойством двойных кавычек.

## Пример

```
<?php  
$capital = 'Москва';  
echo 'Столица России – ', $capital, '<br>';  
?>
```

```
<?php  
$capital = "Москва";  
echo " Столица России – $capital <br>";  
?>
```

## **Строковые операторы**

### **Конкатенация строк**

Оператор конкатенации '.' используется для объединения  
нескольких строк

## Пример

```
<?php  
    $a = "Привет ";  
    $b = $a."мир!";  
    echo $b;  
?>
```



# **Функции для работы со строками**

1. **strlen()** – получает длину строки

**Пример**

```
<?php
```

```
$x = "Привет";
```

```
echo strlen($x);
```

```
echo "<br>";
```

```
echo strlen("Hello");
```

```
?>
```

2. **trim()** удаляет пробельные или другие символы в начале и конце строки.
3. **ltrim()** удаляет пробельные или другие символы в начале строки
4. **rtrim()** удаляет пробельные или другие символы в конце строки

## Пример

```
<?php
$abs=" Пробелы ";
$abs1=trim($abs);
echo "$abs1";
echo "<br>";
$url = '/saitsozдание.ru/php/trim-funktsiya-podrobno.html/';
$url_parts = trim($url, '/');
print_r($url_parts);
?>
```

5. `strip_tags()` удаляет из строки все HTML-теги, за исключением указанных во втором параметре

### Пример

```
<?php
```

```
echo $str = '<p style="color:red"><b>Строка</b></p>';
```

```
echo $str1 = strip_tags($str);
```

```
echo $str2 = strip_tags($str, '<b>');
```

```
?>
```

**6. explode()** разделяет строку на подстроки по указанному разделителю и добавляет их в массив.

### **Пример**

```
<?php
$str = "Фамилия Имя Отчество Год рождения";
$Mass = explode(" ", $str);
foreach ($Mass as $key) {
echo $key . '<br>';
}
?>
```

**7. substr()** возвращает подстроку указанной длины, начиная с заданной позиции.

**substr(<Строка>, <Начальная позиция>, [<Длина>]);**

## Пример

```
<?php  
echo $rest = substr("abcdef", 1);  
echo '<br>';  
echo $rest = substr("abcdef", 0, 2);  
echo '<br>';  
echo $rest = substr("abcdef", -2, 2);  
echo '<br>';  
?>
```



**К отдельным символам можно обращаться с помощью фигурных скобок**

```
$string = 'abcdef';  
echo $string{3};
```

**8. wordwrap()** позволяет разбить длинный текст на строки указанной длины

**wordwrap(<Строка>, <Количество символов>, <Символ разрыва>);**

### **Пример**

```
$str = "Очень длинная строка перед выводом";  
echo wordwrap($str, 7, "<br>");
```

9. **strtoupper()** заменяет все символы строки соответствующими прописными буквами;
10. **strtolower()** заменяет все символы строки соответствующими строчными буквами.

## Пример

```
<?php  
$fio="Hello";  
$fio=strtoupper($fio);  
echo $fio."<br>";  
$fio=strtolower($fio);  
echo $fio;  
?>
```

## Функции для работы с символами

1. **chr(<Код символа>)** возвращает СИМВОЛ ПО указанному коду;
2. **ord(<Символ>)** возвращает код указанного символа.

## Поиск и замена в строке

1. `strpos()` – ищет подстроку в строке.

**Формат:**

```
strpos(<Строка>, <Подстрока>, [<Начальная позиция  
поиска>]);
```

**Пример**

```
<?php
```

```
$str=strpos("Hello PHP", "PHP");
```

```
if ($str!== false) echo "Строка PHP найдена в исходной строке  
в позиции $str";
```

```
else echo "Не найдено";
```

```
?>
```

**2. `str_replace()`** производит замену всех вхождений подстроки в строку на другую подстроку и возвращает результат в виде новой строки.

**Формат:**

**`str_replace(<Подстрока для замены>, <Новая подстрока>, <Строка>, [<Количество произведенных замен>]);`**

## Пример

```
$str = 'Привет, Петя';
```

```
$count = 0;
```

```
$str = str_replace('Петя', 'Вася', $str, $count);
```

```
echo $str;
```

```
echo $count;
```



## Сравнение строк

### Операторы сравнения строк

**Не рекомендуется использовать операторы сравнения == и !=, поскольку они требуют преобразования типов.**

**Оператор эквивалентности === позволяет корректно сравнивать строки**

## Пример

```
$x="123";
```

```
$y=123;
```

```
$z="123";
```

```
if ($x === $z) echo "<p>Строка x равна строке z</p>";
```

```
if ($x === $y) echo "<p>Строка x равна строке y</p>";
```

```
if ($x !== $y) echo "<p>Строка x НЕ равна строке z</p>";
```

## Функции сравнения строк

1. `strcmp(<Строка1>, <Строка2>)` сравнивает две строки.

**Возвращает значения:**

- ✓ 0 – если строки равны;
- ✓ 1 – если <Строка1> больше <Строки2>;
- ✓ -1 – если <Строка1> меньше <Строки2>.

2. `strcasestr(<Строка1>, <Строка2>)` сравнивает две строки без учета регистра

## Пример

```
$str1 = "Stroka";
```

```
$str2 = "stroka";
```

```
echo strcmp($str1, $str2);
```

```
echo strcasecmp($str1, $str2);
```

## Кодирование строк

1. **urlencode()** выполняет URL-кодирование строки

### Пример

```
$str = "Текст на русском языке";  
echo urlencode($str);
```

2. **urldecode()** раскодирует строку, закодированную с помощью функции **urlencode()**

### Пример

```
$str = "Текст на русском языке";  
echo urlencode($str);  
echo '<br>';  
echo urldecode($str);
```

## Кодирование строк

- 3. md5()** – кодирует строку, используя алгоритм MD5.  
Используется для кодирования паролей
- 4. crypt()** – кодирует строку, используя алгоритм DES

### Пример

```
<?php
$pass = "Пароль";
echo $pass.'<br>';
echo md5($pass).'<br>';
echo crypt($pass);
?>
```

## Пример

```
$pass = "password";  
$pass = md5($pass);  
echo $pass;  
$pass2 = "password";  
if ($pass === md5($pass2)) echo "Пароль правильный";
```

## Преобразование кодировок

1. Функция `convert_cyr_string()` преобразует строку из одной кодировки в другую.

`convert_cyr_string(<Исходная строка>, <Исходная кодировка>, <Нужная кодировка>);`

Параметры `<Исходная кодировка>` и `<Нужная кодировка>` могут принимать значения:

- ✓ a или d — кодировка x-cp866;
- ✓ i — кодировка iso8859-5;
- ✓ k — кодировка KOI8-R;
- ✓ m — кодировка x-mac-cyrillic;
- ✓ w — кодировка windows-1251 (cp1251).



**2. Функция `iconv()` также преобразовывает символы строки из одной кодировки в другую.**

**`iconv(<Исходная кодировка>, <Нужная кодировка>[<Флаг>], <Исходная строка>);`**

## Функции для обработки мультбайтных строк

**1. `mb_strlen` (строка [, кодировка])** - возвращает количество символов в строке;

Многобайтный символ вычисляется как 1.

**2. `mb_convert_encoding()`** – конвертирует кодировку СИМВОЛОВ;

**`mb_convert_encoding(<Исходная строка>, <Нужная кодировка>, <Исходная кодировка>);`**

**3. `strlen(<Строка>)`** возвращает количество байт в строке;

## Функции для обработки мультбайтных строк

4. `mb_substr()` возвращает подстроку указанной длины, начиная с заданной позиции.

```
mb_substr(<Строка>, <Начальная позиция>[,  
<Длина>[,<Кодировка>]]);
```

### Пример

```
$str = 'Строка';  
$str1 = mb_substr($str, 0, 1);  
echo $str1;
```

**5. iconv\_substr()** возвращает подстроку указанной длины, начиная с заданной позиции

**iconv\_substr(<Строка>, <Начальная позиция>[, <Длина>[,<Кодировка>]]);**

**Пример**

```
$str = 'Строка';
```

```
$str1 = iconv_substr($str, 0, 1, 'UTF-8');
```

```
echo $str1;
```

**6. `mb_strtoupper(<Строка>[, <Кодировка>]`)** заменяет все символы строки соответствующими прописными буквами

### **Пример**

```
$str = 'очень длинная строка';  
echo mb_strtoupper($str, 'UTF-8');
```

**7. mb\_strtolower(<Строка>[, <Кодировка>])** заменяет все символы строки соответствующими строчными буквами

### **Пример**

```
$str = 'ОЧЕНЬ длинная строка';  
echo mb_strtolower($str, 'UTF-8');
```

8. **mb\_convert\_case(<Строка>, <Режим>[, <Кодировка>])** преобразует регистр символов в зависимости от значения второго параметра.

**Параметр <Режим> может принимать значения:**

- ✓ **MB\_CASE\_UPPER** – заменяет все символы строки соответствующими прописными буквами;
- ✓ **MB\_CASE\_LOWER** – заменяет все символы строки соответствующими строчными буквами;
- ✓ **MB\_CASE\_TITLE** – делает первые символы всех слов прописными.

## Примеры

```
$str = 'ОЧЕНЬ длинная строка';
```

```
echo mb_convert_case($str, MB_CASE_UPPER, 'UTF-8');
```

```
echo '<br>';
```

```
echo mb_convert_case($str, MB_CASE_LOWER, 'UTF-8');
```

```
echo '<br>';
```

```
echo mb_convert_case($str, MB_CASE_TITLE, 'UTF-8');
```



## Функции для поиска в строке

1. `mb_strpos()` ищет подстроку в строке

`mb_strpos(<Строка>, <Подстрока>[, <Начальная позиция поиска>[,<Кодировка>]]);`

2. `mb_stripos()` ищет подстроку в строке, не зависит от регистра символов

`mb_stripos(<Строка>, <Подстрока>[, <Начальная позиция поиска>[,<Кодировка>]]);`

Функции возвращают номер позиции, с которой начинается вхождение подстроки в строку.

## Пример

```
echo mb_strpos('Привет', 'ри', 0, 'UTF-8');  
mb_internal_encoding('UTF-8');  
if (mb_strpos('Привет', 'При') !== false) echo 'Найдено';  
else echo 'Не найдено';
```

3. `mb_strrpos()` ищет подстроку в строке  
`mb_strrpos(<Строка>, <Подстрока>[, <Начальная  
позиция поиска>[,<Кодировка>]]);`

4. `mb_strripos()` ищет подстроку в строке  
`mb_strripos(<Строка>, <Подстрока>[, <Начальная  
позиция поиска>[,<Кодировка>]]);`

Возвращают позицию последнего вхождения подстроки в строку

`mb_strripos()` не зависит от регистра символов.

5. **iconv\_strpos()** ищет подстроку в строке. Возвращает номер позиции, с которой начинается вхождение подстроки в строку.

**iconv\_strpos(<Строка>, <Подстрока>[, <Начальная позиция поиска>[,<Кодировка>]]);**

6. **iconv\_strrpos()** ищет подстроку в строке. Возвращает позицию последнего вхождения подстроки в строку.  
**iconv\_strrpos(<Строка>, <Подстрока>[, Кодировка]);**

Функции зависят от регистра символов.

**7. mb\_substr\_count()** возвращает число вхождений подстроки в строку. Функция зависит от регистра символов.  
**mb\_substr\_count(<Строка>, <Подстрока>[,<Кодировка>]);**

Параметр <Кодировка> во всех функциях является  
необязательным.