Preparação para Maratona de Informática
"PHP"

Neste material veremos como abrir arquivos de texto utilizando algumas bibliotecas PHP para manipulação de arquivos, conceitos básicos de manipulação de arrays (vetores), formatação de saída e algumas outras funções.

Como referência, todos arquivos .TXT utilizados devem estar na mesma pasta onde estiver o *script* .PHP.

Abertura de arquivos

Para abrir um arquivo de texto, tanto para leitura tanto para escrita, usa-se a função `fopen(filename, mode)`, já atribuindo seu conteúdo a uma variável, desta forma:

```
$filename = "xanadu.txt";
$content  = fopen($filename, 'r');
```

Neste caso, usamos a opção "r", que abre o arquivo somente para leitura.

```
$filename = "outagain.txt";
$fileopen = fopen($filename, 'a');
```

Neste caso, usamos a opção "a", que abre o arquivo somente para gravação e escreve no final do arquivo.

Os valores possíveis para este parâmetro são:

- "r" (Somente leitura. Inicia com o ponteiro no início do arquivo)
- "r+" (Leitura e gravação. Inicia com o ponteiro no início do arquivo)
- "w" (Somente gravação. Abre e limpa o conteúdo do arquivo; ou cria um novo arquivo, caso ele não exista)
- "w+" (Leitura e gravação. Abre e limpa o conteúdo do arquivo; ou cria um novo arquivo, caso ele não exista)
- "a" (Somente gravação. Abre a passa a gravar no final do arquivo ou cria um novo arquivo, caso ele não exista)

- "a+" (Leitura e gravação. Preserva o conteúdo inicial, gravando no final do arquivo)
- "x" (Somente gravação. Cria um novo arquivo. Retorna FALSE e um erro se o arquivo já existe)
- "x+" (Leitura e gravação. Cria um novo arquivo. Retorna FALSE e um erro se o arquivo já existe)

Deve-se verificar a permissão de acesso à pasta e aos arquivos para que seja feita a manipulação solicitada. Para gravar em arquivos e em pastas, é necessário que eles estejam com atributo de leitura e gravação. Normalmente, estas permissões são abertas em Windows e nada precisa ser feito; em servidores Linux, por padrão as pastas não têm permissão de escrita. Pesquise sobre "CHMOD" para servidores Linux e "permissões de acesso a pastas" para servidores Windows para obter mais informações.

Lendo o conteúdo do arquivo

Como o arquivo foi aberto e seu conteúdo já atribuído a uma variável, basta usar um `while` para ler cada parágrafo do arquivo. Se for um arquivo de dados, por exemplo, cada linha é um parágrafo, delimitado por uma marcação de quebra de linha.

```
$i = 0;
while (!feof($content)) {
  $conteudo[$i] = fgets($content);
  $i++;
}
```

Desta forma, tem-se cada linha dentro de uma posição do vetor `$conteudo`. A partir daí, pode-se usar o vetor da maneira que for necessário.

Se for um banco de dados, é importante quebrar as colunas de cada linha com o marcador que estiver sendo utilizado. Caso seja vírgula, como no caso de arquivos separados por vírgula, pode ser utilizado um laço `for` para utilizar cada informação de cada campo.

```
echo "<table>";
for ($x = 0; $x < count($conteudo); $x++) {
  $campo = explode(",", $conteudo[$x]);
  echo "<tr>";
  for ($y = 0; $y < count($campo); $y++) {
    echo "<td>" . $campo[$y] . "</td>";
  }
  echo "</tr>";
}
echo "</table>";
```

No exemplo acima, foi montada uma tabela com os dados contidos no arquivo. A função `count(vetor)` conta quantas posições em um vetor, e foi utilizada como limite do laço.

No próximo passo deste treinamento, será melhor explicada a manipulação de arrays (vetores) em PHP.

Escrevendo conteúdo no arquivo

A função `fwrite(arquivo, texto)` grava um texto no arquivo aberto. Dependendo da opção dada, o ponteiro ficará no início ou no fim do documento. Além disso, obviamente, devem ser usadas as opções que dão direito a gravação no arquivo (r+, w+, a e x+). Será abordado este assunto com mais detalhes no passo 4 deste treinamento.

```
$filename = "outagain.txt";
$fileopen = fopen($filename, 'a');
fwrite ($fileopen, "Texto a ser escrito no arquivo");
```

Sempre fechar os arquivos

Utilizar sempre o `fclose(arquivo)` no final do uso de cada arquivo para que, tanto o próprio sistema de entrada quanto o de saída, possam acessar posteriormente o arquivo pelo sistema operacional ou outros programas.

```
fclose($fileopen);
fclose($content);
```

Utilização na Maratona

Há diversas maneiras de estruturar textos de entrada e saída através de caracteres e linhas. Veremos mais sobre o assunto nas próximas etapas do treinamento.

Durante a Maratona, os alunos receberão exemplos de arquivos de textos de entradas, que deverão ser processadas para a solução dos problemas apresentados, gerando arquivos de textos de saídas, com as soluções encontradas.

Em cada problema apresentado, os nomes dos arquivos de entradas e de saídas, são fornecidos.

Arquivo para consulta (arquivo.php)

```
$filename = "xanadu.txt";
$content  = fopen($filename, 'r');

$i = 0;
while (!feof($content)) {
  $conteudo[$i] = fgets($content);
  $i++;
}

echo "<table>";
for ($x = 0; $x < count($conteudo); $x++) {
  $campo = explode(",", $conteudo[$x]);
  echo "<tr>";
  for ($y = 0; $y < count($campo); $y++) {
    echo "<td>" . $campo[$y] . "</td>";
  }
  echo "</tr>";
}
echo "</table>";

$filename = "outagain.txt";
$fileopen = fopen($filename, 'a');
fwrite ($fileopen, "Texto a ser escrito no arquivo");

fclose($fileopen);
fclose($content);
```

Vetores (unidimensionais)

Vetor é uma estrutura que necessita apenas de um índice para a identificação de um elemento nele contido. Sendo assim, para manipular um valor em um vetor é necessário fornecer o nome (identificador) do vetor e o índice do elemento desejado. O índice determina a posição na estrutura onde o elemento está inserido. Cada posição de um vetor contém exatamente um valor que pode ser manipulado individualmente.

Declaração: A declaração de um vetor deverá ser escrita da seguinte forma:

```
$vetor = array();
```

Para criar um vetor com elementos já preenchidos, pode-se fazer da seguinte forma:

```
$vetor1 = array("A","B","C","D","E","F","G","H");
$vetor2 = array(1,2,3,4,5,6,7,8,9,10);
```

Cada um dos elementos internos deve ser do mesmo tipo e a variável que armazena o array terá o tipo dos dados atribuídos a ela. O exemplo acima cria um array de Strings que contém 8 elementos chamado `$vetor1` e um vetor de inteiros que contém 10 elementos ambos chamado `$vetor2`.

Acessando os valores de um array

Uma vez que você tem um array com valores iniciais, você pode testar e mudar os valores em cada índice de cada array.

Os arrays em PHP sempre iniciam na posição 0 e têm a quantidade de posições dinâmica, ou seja, o vetor aceita a quantidade de valores que for necessária e sempre estará preenchido até a última posição. Por exemplo:

```
$vetor1[0] = "A";
$vetor1[1] = "B";
```

Caso seja referenciada uma posição que não foi atribuído nenhum valor, ele estará vazio e o máximo que acontecerá é que o Apache (servidor de páginas PHP) retornará um Warning avisando que a variável naquela posição não foi inicializada. No caso de servidores remotos, normalmente, eles estão configurados para não exibirem erros do nível Warning.

Obtendo o tamanho de um array

```
$tamanho1 = count($vetor1); //retornará o valor 8
$tamanho2 = count($vetor2); //retornará o valor 10
```

A forma de acesso a um determinado elemento do array é feita simplesmente fazendo referência ao seu índice.

O acesso a um elemento do array é feito colocando uma expressão de valor inteiro entre colchetes após o nome do array.

6.0	7.0	9.0	5.5	9.1	10.0	4.7	7.5	8.6	8.0	nota
0	1	2	3	4	5	6	7	8	9	

A variável `$nota[3]` faz referência ao elemento do vetor, cujo conteúdo é 5.5.

Desta forma, para se ter acesso a qualquer uma das notas armazenadas basta utilizar uma variável inteira qualquer (a título de exemplo a variável `$i`) como sendo o índice. Supondo `$i = 5`, uma referência a `$nota[$i]`, `$i` seria substituído pelo seu conteúdo no dado instante e neste caso, o valor do elemento referenciado é 10.0.

O exemplo a seguir cria um array de inteiros chamado tecladoNum e depois preenche o array com inteiros de 0 a 9:

```
$tecladoNum;
for ($i = 0; $i < count($tecladoNum); $i++)
    $tecladoNum[$i] = $i;
```

Exemplo 1

Dado um conjunto com os seguintes elementos: 1, 2, 4, 8, 6, 7, 15, 9, 10, 18. Elabore um aplicativo para calcular a soma desses elementos. (arquivo somavet.php)

```
$a = array(1,2,4,8,6,7,15,9,10,18);
$total = 0;
$saida = "Elementos do vetor:<br>";
for ($i = 0; $i < count($a); $i++) {
  $saida .= $a[$i] . " ";
  $total += $a[$i];
}
echo $saida . "<br>Soma dos elementos do vetor " . $total;
```

Exemplo 2

Elabore um aplicativo para que o usuário preencha um vetor de 10 elementos e os exiba logo depois. (arquivo exibevet.php)

```
//verifica se o formulário já foi submetido
if (!isset($_POST['ok'])) {
  //se não foi, monta o formulário com um campo para cada valor
  echo "<form action='' method='post'>";
  for ($i = 0; $i < 10; $i++) {
    echo "Valor " . $i . ": ";
    echo "<input type='text' name='valor[" . $i . "]' size='1'>";
    echo "<br>";
  }
  echo "<input type='submit' value='Ok' name='ok'>";
  echo "</form>";
}
else {
  //se os dados já foram submetidos, exibe-os
  for ($i = 0; $i < 10; $i++) {
    echo "Valor " . $i . ": " . $_POST['valor'][$i] . "<br>";
  }
}
```

Importante lembrar que os dados enviados pelo método post do formulário serão armazenados em um vetor chamado `$_POST`.

Exemplo 3

Elabore um aplicativo para que o usuário preencha um vetor de 10 elementos, os exiba logo depois e permita que o usuário verifique o elemento pela posição. (arquivo valores.php)

```
//verifica se nenhum formulário foi submetido
if (!isset($_POST['ok']) && !isset($_POST['posicao'])) {
  //se não foi, monta o formulário com um campo para cada valor
  echo "<form action='' method='post'>";
  for ($i = 0; $i < 10; $i++) {
    echo "Valor " . $i . ": ";
    echo "<input type='text' name='valor[" . $i . "]' size='1'>";
    echo "<br>";
  }
  echo "<input type='submit' value='Ok' name='ok'>";
  echo "</form>";
}
else if (!isset($_POST['posicao'])) {
  //se o vetor foi preenchido, mas a posição ainda não, exibe-os
  for ($i = 0; $i < 10; $i++) {
    echo "Valor " . $i . ": " . $_POST['valor'][$i] . "<br>";
  }
  //cria o formulário para solicitar a posição a ser consultada
  echo "<form action='' method='post'>";
  echo "Posição a ser consultada: ";
  echo "<input type='text' name='posicao' size='1'>";
  //reenvia os dados por POST, em campos ocultos
  for ($i = 0; $i < 10; $i++) {
    echo "<input type='hidden' name='valor[" . $i . "]'";
    echo " value='" . $_POST['valor'][$i] . "'>";
  }
  echo "<br>";
  echo "<input type='submit' value='Ok' name='ok'>";
  echo "</form>";
}
else {
  $p = $_POST['posicao'];
  echo "O elemento na posição " . $p . " é " . $_POST['valor'][$p];
}
```

Formatando saída numérica

O método utilizado em PHP para formatação de valores numéricos é `number_format(valor,casas_decimais,separador_dec,separador_milhar)`, que envia para a saída um simples valor após convertê-lo da maneira apropriada. Podemos ver o que ocorre no exemplo a seguir:

```
$i = 2;
$r = sqrt($i);
echo "The square root of " . $i . " is " . $r . "<br>";

$i = 5;
$r = sqrt($i);
echo "The square root of " . $i . " is " . $r . "<br>";
```

A saída será:

The square root of 2 is 1.4142135623731
The square root of 5 is 2.2360679774998

As variáveis `$i` e `$r` não foram formatadas.

O Método number_format

O método `number_format()` formata números baseado numa série de parâmetros de formatação. O primeiro argumento é o valor a ser formatado; o segundo, a quantidade de casas decimais; o terceiro, o separador de decimais e o quarto, o separador de milhares. A string formatada não sofre alterações no seu conteúdo, apenas o que será mostrado na saída padrão é que sairá formatado. Caso o seu retorno seja atribuído a uma variável, a nova variável armazenará o número com o resultado da formatação. (arquivo format.php)

```
$i = 2;
$r = sqrt($i);
echo "The square root of " . $i . " is " . $r . "<br>";

$i = 5;
$r = sqrt($i);
echo "The square root of " . $i . " is " . $r . "<br>";

$s = number_format($r, 4, ",", ".");
echo "Número formatado: " . $s;
```

Aqui está a saída:

The square root of 2 is 1.4142135623731
The square root of 5 is 2.2360679774998
Número formatado: 2,2361

Somando-se as possibilidades de conversão, as especificações de formatos podem conter diversos elementos adicionais que permitirão customizar a saída formatada. Consulte a função `number_format()` em php.net, além de outras funções de manipulação e formatação de strings de saída.

Método explode()

O método `explode(regex, texto, [limite])` quebra a string de acordo com a expressão regular dada, onde o regex é a expressão regular delimitadora, o texto é a sequência de caracteres (string) a ser manipulada e o limite é o controlador de vezes que ela deve ser aplicada, conforme explicação mais pormenorizada abaixo.

O array de retorno desse método contém cada substring desta string terminada com outra substring que seja igual à expressão regular fornecida ou que termine com ela. As substrings no array gerado estarão na mesma ordem que aparecem na string original. Se a expressão regular fornecida não for igual à nenhuma parte da string original, o array resultante conterá apenas um elemento com todos os caracteres da string original.

O parâmetro reges deve ser escrito em expressão regular e pode conter vários caracteres em um array para que sirvam de delimitadores. Neste caso, serão utilizados todos entre colchetes: vírgula (,), ponto-e-vírgula (;) e *pipe* (|).

```
$campos = explode([";",",";"|"], $conteudo);
```

O parâmetro de limite (opcional) controla o número de vezes que o padrão da expressão regular fornecida será aplicado na string original e afetará a quantidade de elementos do array resultante. Se o limite *n* for maior que zero então o padrão será aplicado em pelo menos *n* - 1 vezes, o tamanho do array não será maior que *n*, e a última entrada conterá todas as entradas que combinarem com o delimitador. Se *n* for negativo, então o padrão será aplicado quantas vezes possível e o array terá um tamanho qualquer. Se *n* for zero, o padrão será aplicado quantas vezes possível, o array terá um tamanho qualquer e qualquer espaço vazio nas strings será descartado.

Aplicando o método `explode()` na string "boo:and:foo", por exemplo, fornecerá os seguintes resultados com estes parâmetros:

Regex	Limit	Result
:	2	{ "boo", "and:foo" }
:	5	{ "boo", "and", "foo" }
:	-2	{ "boo", "and", "foo" }
o	5	{ "b", "", ":and:f", "", "" }
o	-2	{ "b", "", ":and:f", "", "" }
o	0	{ "b", "", ":and:f" }

Retorno:

Um array de strings formado pela “quebra” da string original em pedaços (substrings) delimitados pela expressão regular.

Leitura de Arquivo e Gravação do Resultado

Exemplo de Leitura de Arquivo utilizando o método `explode()` para separação de dados das linhas, já abordado no primeiro passo deste treinamento. (arquivo arquivo2.php):

```
$filename = "xanadu.txt";
$content  = fopen($filename, 'r');

$filename = "outagain.txt";
$fileopen = fopen($filename, 'w');

$i = 0;
while (!feof($content)) {
  $conteudo[$i] = fgets($content);
  $i++;
}

echo "<table>";
for ($x = 0; $x < count($conteudo); $x++) {
  $campo = explode([";",",","|"], $conteudo);
  echo "<tr>";
  for ($y = 0; $y < count($campo); $y++) {
    echo "<td>" . $campo[$y] . "</td>";
    fwrite($fileopen, $campo[$y]);
    if ($y < count($conteudo)-1)
      fwrite($fileopen, ",");
    else
      fwrite($fileopen, "\n");
  }
  echo "</tr>";
}
echo "</table>";

fclose($fileopen);
fclose($content);
```

Perceba que o programa lê um arquivo de texto, composto de algumas linhas, com itens separados pelos caracteres apontados no primeiro parâmetro.

As linhas são “quebradas” nos seus elementos, de acordo com o posicionamento das dos delimitadores e esses elementos são exibidos sequencialmente, linha a linha, em uma tabela. Logo depois, os dados são escritos no arquivo de saída. Entre cada campo, insere uma vírgula (se o contador estiver até o penúltimo campo, `$y < count($conteudo)`)

e, a cada final de linha (se o contador não estiver até o penúltimo campo, está no último), insere o caracter "\n", para que haja a quebra da linha.

Este exemplo poderia ser usado para trocar qualquer tipo de separadores (neste caso, vírgula, ponto-e-vírgula ou *pipe*) usados no arquivo de origem, por vírgula no arquivo de destino.

Deve-se verificar a permissão de acesso à pasta e aos arquivos para que seja feita a manipulação solicitada. Para gravar em arquivos e em pastas, é necessário que eles estejam com atributo de leitura e gravação. Normalmente, estas permissões são abertas em Windows e nada precisa ser feito; em servidores Linux, por padrão as pastas não têm permissão de escrita. Pesquise sobre "CHMOD" para servidores Linux e "permissões de acesso a pastas" para servidores Windows para obter mais informações.

Esse será o padrão dos exercícios que serão propostos aos times de alunos na Maratona de Programação.