

4.2 Programação em PHP: elementos básicos de programação

CET - Desenvolvimento de Produtos
Multimédia

António Couto, Tânia Alves e Luís Mendes Gomes

Sumário

- Tipos de Dados (Tabelas...).
- Variáveis (ligação com o HTML).
- Controlo de execução.
- Funções (Localidade das Variáveis).
- Constantes.
- Variáveis do Ambiente.

Tipos de Dados

- Primitivos
 - Valores lógicos (TRUE, FALSE)
 - Números inteiros (`$num_int = 2`; `$num_int = 0x12`)
 - Números em vírgula flutuante (`$num_fl = 4.5`; `$num_fl = 7E - 10`)
 - Cadeias de caracteres (`$str = "Olá, $nome"`; `$str = 'Olá, $nome'`)
- Compostos
 - Tabelas (`$tab = array (2, 3.141692, "António", array(3, "Pedro"), "Tânia")`
`($tab = array (array (1,"Tânia"),`
`array ("António", 2),`
`array (3.141692, TRUE)))`
 - Objectos (Variável de um tipo de dados criado pelo utilizador que pode conter múltiplos atributos, definida através de um método construtor)
- Especiais
 - Resource (referências para recursos externos: bases de dados, ficheiros, sockets...)
 - Null (objecto nulo)

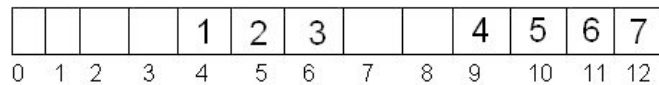
Vectores Associativos

```
$tab = array (4=> 1, 2, 3, 9=> 4,5,6,7);
```

```
echo $tab[0]; # produz erro
```

```
echo $tab[4]; # imprime 1
```

```
echo $tab[5]; # imprime 2
```

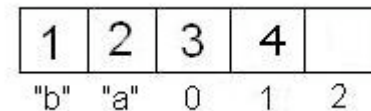


```
$tab = array ("b"=>1, "a"=>2, 3, 4);
```

```
echo $tab["a"]; # imprime 2
```

```
echo $tab[0]; # imprime 3
```

```
echo $tab[2]; # produz erro
```

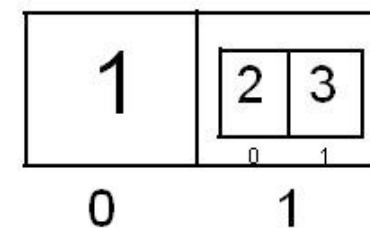


```
$tab = array (1, array (2,3));
```

```
echo $tab[1][0]; # imprime 2
```

```
echo $tab[0]; # imprime 1
```

```
echo $tab[0][0]; # produz erro
```



Interacção com o utilizador via HTTP

- método `get` - associado ao URL de um documento PHP; variável (do tipo array) `$_get`.
- método `post` - enviado numa mensagem HTTP; variável (do tipo array) `$_post`.

Operadores e Expressões

- Aritméticos (+ (adição), - (subtração), * (multiplicação), / (divisão), % (resto da divisão inteira))
- Cadeias de caracteres (. (concatenação))
- Atribuição (=, +=, -=, *=, ./, ++, --, ...)
 - Ex: \$a += \$b; # é o mesmo que \$a = \$a + \$b
 - \$a++; # é o mesmo que \$a = \$a + 1
- Bits (&, |, <<, >>, ...) (operações bit a bit)
 - Ex: \$a << \$b; # desloca \$a para a esquerda \$b bits
- Comparação (== (igual), != ou <> (diferente), === (idêntico), !== (não idêntico), < (menor), > (maior), ...)
 - Ex: \$a=5; \$b=5.0; # \$a === \$b retorna FALSE, \$a == \$b retorna TRUE
 - \$a=5; \$b=6; # \$a === \$b retorna FALSE, \$a == \$b retorna FALSE
 - \$a=6; \$b=6; # \$a === \$b retorna TRUE, \$a == \$b retorna TRUE
- Lógicos (! (negação), && ou and (conjunção), || ou or (disjunção exclusiva))

1	2	3	4	5	6
0	1	2	3	4	5
- Concatenação de vectores (+)
 - Ex: \$a = array(1,2,3); \$b = array(4,5,6); \$c = \$a + \$b; # o array \$c é igual a
- Execução (` `)(executar programas externos)

Comentários

Podem ser escritos de 3 maneiras diferentes:

Para escrever em apenas uma linha: // ou #;

Para escrever um bloco de texto: iniciar com /* e terminar com */.

Controlo da execução

Instrução `if`

`if` (condição)

`bloco_de_instruções` (*)

```
if ( $a > $b )  
{  
    echo "$a é maior que $b";  
}
```

(*) Um bloco de instruções é definido como uma sequência de instruções separadas por `;` e delimitado por `{` à esquerda e `}` à direita. Se apenas existir uma instrução, não existe a necessidade de delimitar esta por chavetas.

Controlo da execução

Instrução `if-else`

```
if (condição)
    bloco_de_instruções_1
else
    bloco_de_instruções_2
```

```
if ( $a > $b )
    echo "$a é maior que $b" ;
else
    echo "$a é menor ou igual $b";
```

Controlo da execução

Instrução `if-elseif`

```
if (condição_1)
    bloco_de_instruções_1
elseif (condição_2)
    bloco_de_instruções_2
...
elseif (condição_n)
    bloco_de_instruções_n
else
    bloco_de_instruções_n+1
```

```
if ( $a > $b)
    echo “$a é maior que $b” ;
elseif ($a = $b)
    echo “$a é igual que $b“;
else
    echo “$a é menor que $b”;
```

Controlo da execução

Instrução switch

```
switch (expressão) {  
    case valor_l:  
        instrução_l;  
        ...  
        instrução_kl;  
        break;  
    ...  
    case valor_n:  
        instrução_l;  
        ...  
        instrução_kn;  
        break;  
    default:  
        instrução_predefinida ;  
}
```

```
switch ($i) {  
    case 0: echo "$i igual a 0";  
        break;  
    case 1: echo "$i igual a 1";  
        break;  
    case 2: echo "$i igual a 2";  
        break;  
    default:  
        echo "$i é maior que 2";  
}
```

Controlo da execução

Instrução for

```
for (inicialização; condição; passo)  
    bloco_de_instruções
```

```
for ( $i = 1; $i <= 10; $i++)  
    echo "$i";
```

Controlo da execução

Instrução `while` e `do-while`

```
while (condição)
    bloco_de_instrucoes
```

```
$i = 1;
while ( $i <= 10 )
    echo $i++;
```

```
do
    bloco_de_instrucoes
while (condição)
```

```
$i = 1;
do {
    echo $i;
    $i++;
} while ( $i <= 10 );
```

Funções

```
function nome (lista_de_parametros) {  
    sequência_de_instruções;  
    return expressao;  
}
```

Passagem por valor:

```
function soma ($n) {  
    $soma=0;  
    for ($i=1;$i<=$n;$i++)  
        $soma+=$i;  
    return $soma;  
}
```

Passagem por Referência:

```
function soma (&$n) {  
    $soma=0;  
    for ($i=1;$i<=$n;$i++)  
        $soma+=$i;  
    return $soma;  
}
```

...
\$valor=soma(5); - atribui o valor à variável
echo soma(5); - mostra o valor

Funções (cont.)

Passagem por omissão:

```
function faz_cafe ($origem, $tipo = 'com leite') {  
    return "Fazer uma chávena de café $origem $tipo. \n";  
}
```

Ao fazer...

```
echo faz_cafe ("de Timor");  
echo faz_cafe ("do Fogo", "expresso");
```

...Obtemos:

```
Fazer uma chávena de café de Timor com leite.  
Fazer uma chávena de café do Fogo expresso.
```

Recursividade

```
function factorial ($n)
  if ($n==0)
    return 0;
  else
    return factorial($n-1)*$n;
```

Visibilidade das variáveis

Visibilidade (local ou global)

`$a=1;` - Variável global

```
function teste()  
    $a=2; - Variável local  
    echo $a;
```

```
$a=1;  
$b=2;  
function soma(){  
    $c=3;  
    global $a, $b;  
    return $a+$b+$c;  
}
```

```
$a=1;  
$b=2;  
function soma(){  
    $c=3;  
    return $GLOBALS["$a"]+$GLOBALS["$b"]+$c;  
}
```

Tipos de variáveis

- **Estáticas**

Variáveis que mantêm o seu valor mesmo após o programa ter abandonado o âmbito onde elas estão definidas.

- **Dinâmicas**

Variáveis com capacidade para guardar o nome de outras variáveis

```
$a='ola';  
$$a=' mundo';
```

\$\$a é o mesmo que escrever \$ola.

```
echo ($a.$ola);  
echo ($a.$$a);
```

Ambas as expressões imprimem
'ola mundo'

Variáveis externas

- `__FILE__` – Contém o nome do programa
- `__LINE__` – Contém a linha da execução do programa
- `PHP_VERSION` – Versão do PHP
- `PHP_OS` – Contém o nome do sistema operativo no qual o PHP está a ser executado
- `E_ERROR` – Representa um erro de processamento sem recuperação
- `E_WARNING` – Representa um erro que não influencia a execução de um programa
- `E_PARSE` – Suspensão definitiva da execução do programa
- `E_NOTICE` – Representa um aviso e a execução do programa pode continuar

Podem ser definidas novas constantes pelo utilizador através da instrução `define`:

```
define (nome, valor);
```

Variáveis externas

- `$GLOBALS` – Contém referência todas as variáveis globais
- `$_GET` – Variáveis enviadas para o programa através do HTTP GET
- `$_POST` - Variáveis enviadas para o programa através do HTTP POST
- `$_COOKIE` - Variáveis enviadas para o programa através dos cookies HTTP
- `$_FILES` - Variáveis enviadas para o programa via transferência de ficheiros HTTP
- `$_REQUEST` - `$_GET + $_POST + $_COOKIE`
- `$_SESSION` – Contem as variáveis na sessão do programa