FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

LẬP TRÌNH PHP1

TỔNG QUAN VỀ PHP

http://www.poly.edu.vn

Mục tiêu

- Mảng (array)
- Hằng (constant)
- Cấu trúc điều khiển
- Cấu trúc lặp
- Break, continue, goto

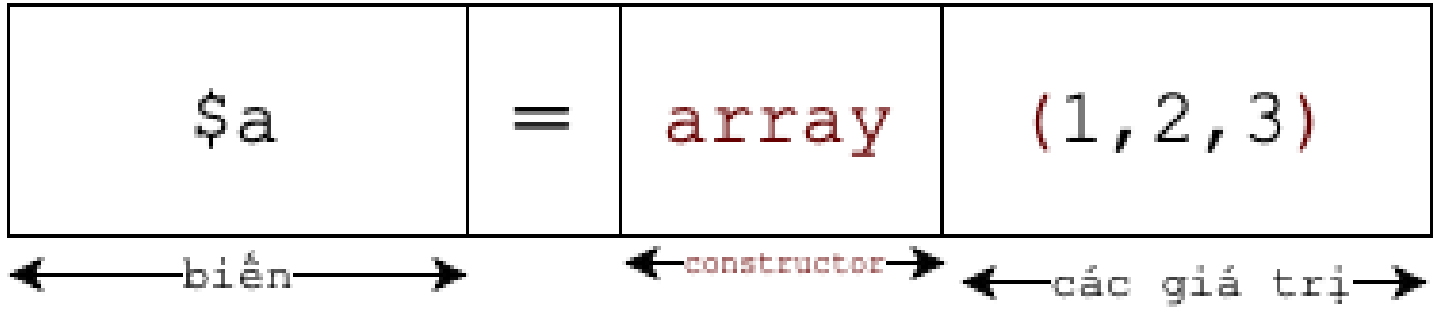
PHẦN 1

MẢNG (ARRAY)

- Mảng được sử dụng để lưu trữ một tập các giá trị trong một biến duy nhất.
- Mảng trong PHP bao gồm cặp ***key-value.***
- ***Key*** (khoá) có thể là một số nguyên (mảng số – numeric array), một chuỗi (mảng kết hợp) hoặc cả hai (mảng hỗn hợp)
- ***Value*** có thể là bất kỳ kiểu dữ liệu nào.

MẢNG SỐ (NUMERIC ARRAY)

- Mảng được tạo bằng cách sử dụng *array* constructor

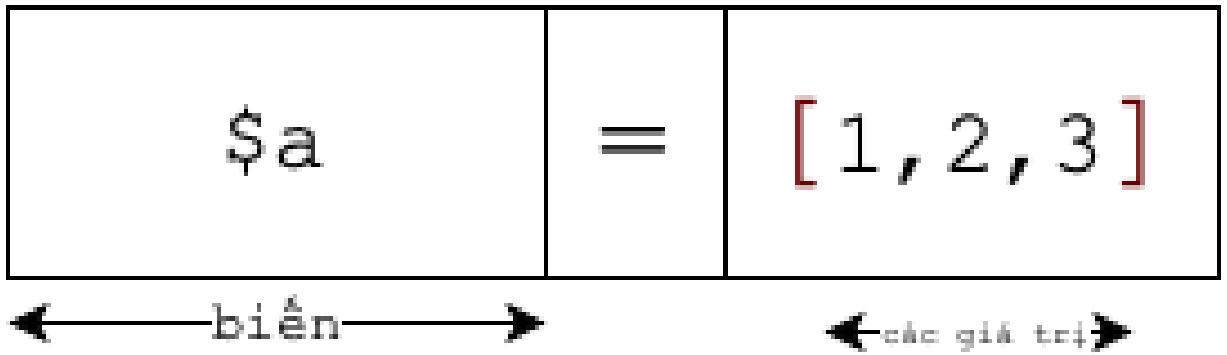


- Ví dụ

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

MẢNG SỐ (NUMERIC ARRAY)

- Trong PHP5.4, có thể khai báo mảng bằng cách ngắn hơn đó là thay thế array constructor bằng cặp ngoặc vuông



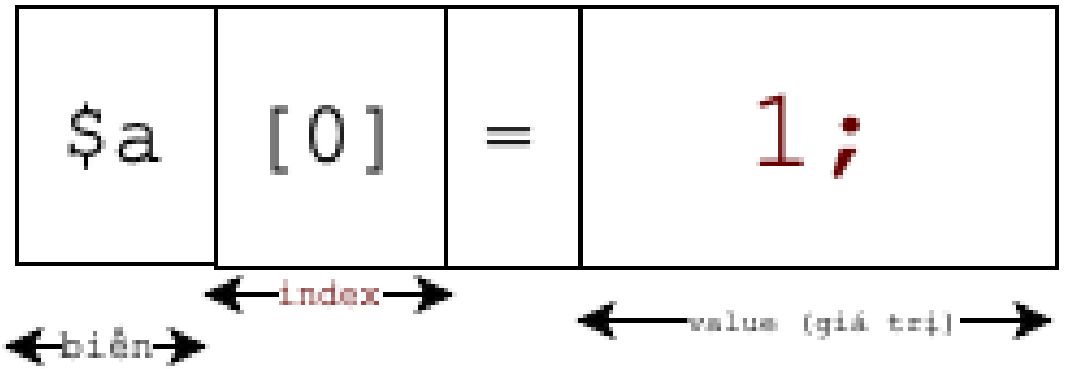
- Ví dụ

```
$a = [1,2,3];
```

MẢNG SỐ (NUMERIC ARRAY)

- Một cách khác để tạo mảng là các phần tử của mảng có thể được tham chiếu bằng cách đặt ***index*** của phần tử mong muốn trong dấu ngoặc vuông

- Lưu ý: nếu là mảng số thì index bắt đầu từ số 0



MẢNG SỐ (NUMERIC ARRAY)

- Ví dụ

```
$a[0] = 1;
$a[1] = 2;
$a[2] = 3;
```

- Thêm phần tử vào mảng

```
$a[3] = 4;
```

- Thêm phần tử vào cuối mảng

```
$a[] = 5; // $a[4]
```

MẢNG SỐ (NUMERIC ARRAY)

- Truy xuất phần tử trong mảng số

```
echo "$a[0] $a[1] $a[2] $a[3]"; // "1 2 3 4"
```

index

Variable
(biến)

MẢNG LIÊN KẾT (ASSOCIATIVE ARRAY)

- Trong mảng liên kết: khoá là một chuỗi thay vì là numeric index. Key chính là cung cấp tên cho phần tử thay vì dùng số.
- Khi tạo mảng liên kết, sử dụng toán tử mũi tên kép double arrow operator (=>) để biết được khoá (key) này tham chiếu đến giá trị (value) nào.

```
$b = array(
    'one' => 'a',
    'two' => 'b',
    'three' => 'c'
);
```

key *value*

MẢNG LIÊN KẾT (ASSOCIATIVE ARRAY)

- Tham chiếu thành phần trong mảng liên kết.

```
$b['one'] = 'a';
$b['two'] = 'b';
$b['three'] = 'c';

echo $b['one'] . $b['two'] . $b['three']; // "abc"
```

MẢNG LIÊN KẾT (ASSOCIATIVE ARRAY)

- Toán tử mũi tên kép cũng có thể được sử dụng để khai báo mảng số

```
$c = array(0 => 0, 1 => 1, 2 => 2);
```

- Hoặc mảng số lược bớt key

```
$e = array(5 => 5, 6);
```

MẢNG HỖN HỢP(MIXED ARRAY)

- PHP không phân biệt mảng liên kết và mảng số, do đó các phần tử của mỗi loại có thể được kết hợp trong cùng 1 mảng.

```
$d = array(0 => 1, 'foo' => 'bar');
```

- Truy xuất phần tử trong mảng hỗn hợp

```
echo $d[0] . $d['foo']; // "1bar"
```

MẢNG NHIỀU CHIỀU(MULTI-DIMENSIONAL ARRAY)

- Mảng nhiều chiều là một mảng có chứa mảng khác.

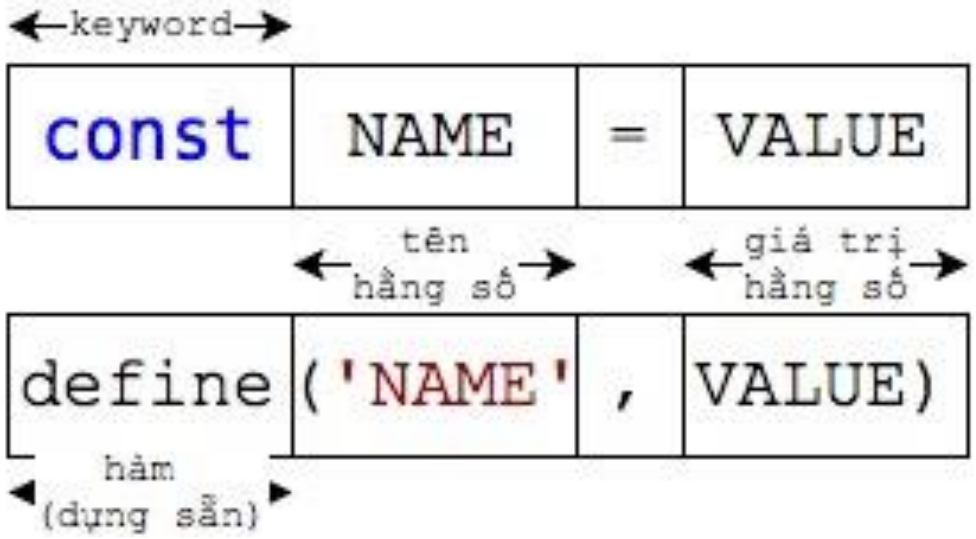
```
$a = array(array('00', '01'), array('10', '11'));

$a[0][0] = '00';
$a[0][1] = '01';
$a[1][0] = '10';
$a[1][1] = '11';
```

- Truy xuất phần tử trong multi-dimension array

```
echo $a[0][0] . $a[0][1] . $a[1][0] . $a[1][1];
```

CONSTANTS(HẰNG)

- Hằng (constant): là một biến với giá trị không thay đổi trong quá trình thực thi.
- PHP cung cấp 2 phương pháp để tạo hằng:
  - ***Const*** modifier
  - ***Define*** function (Hàm define)



CONSTANTS(HẰNG)

- ***Const*** modifier không được áp dung cho biến và tham số cục bộ (local variables - parameters). Từ PHP5.3, const được sử dụng để tạo hằng toàn cục (global constants). Hằng được xác định trong phạm vi toàn cục và có thể được truy cập bất kỳ đâu trong tập lệnh sau khi nó được xác định.

- Ví dụ:

```
const PI = 3.14;
echo PI; // "3.14"
```

CONSTANTS(HẰNG)

- ***Define*** function có thể tạo hằng toàn cục và cục bộ.

```
define('DEBUG', 1);
```

- Truy xuất hằng không cần sử dụng dấu $ trước tên hằng.

```
echo DEBUG; // "1"
```

- Giá trị hằng sử dụng hàm define có thể là bất kỳ kiểu dữ liệu nào: int, float, string hoặc bool.
- Hàm define cho phép xác định hằng là một biểu thức

```
define('ONE', 1);     // 1
define('TWO', ONE+1); // 2
```

CONSTANTS(HẰNG)

- ***Hằng số*** có phân biệt chữ hoa thường. Tuy nhiên, nếu them đối số thứ 3 là TRUE vào hàm define để tạp một hang số không phân biệt chữ hoa, thường.

```
define('DEBUG', 1, true);
echo debug; // "1"
```

- Để kiểm tra hằng đã tồn tại hay chưa, ta dùng hàm *defined* (làm việc với *const* hoặc *define*)

```
if (!defined('PI'))
  define('PI', 3.14);
```

CONSTANTS(HẰNG)

- Trong PHP7, có thể tạo 1 hằng là mảng.

```
const CA = [1, 2, 3];     // PHP 5.6 or later
define('DA', [1, 2, 3]); // PHP 7 or later
```

CONDITIONALS (CẤU TRÚC ĐIỀU KHIỂN)

- Các câu lệnh điều kiện được sử dụng để thực thi các khối mã khác nhau dựa trên các điều kiện khác nhau.
- Cấu trúc điều kiện If

```php
<?php

if ( condition1 ){
    // Các câu lệnh
} else if( condition1 || condition2 ){
    // Các câu lệnh
} else {
    // Các câu lệnh
}

?>
```

```php
if( condition1 ):
    //block code
elseif ( condition2 ):
    //block code
else:
    //block code
endif;
```

CONDITIONALS (CẤU TRÚC ĐIỀU KHIỂN)

- Ví dụ

```
if ($x == 1)
  echo 'x is 1';
elseif ($x == 2)
  echo 'x is 2';
else
  echo 'x is something else';
```

CONDITIONALS (CẤU TRÚC ĐIỀU KHIỂN)

Switch…case

Sử dụng để kiểm tra một giá trị và đưa ra quyết định dựa trên kết quả của việc so khớp giá trị.

Cú pháp:

```php
switch( $variable ){
    case value:
        //Statement
    break;

    case value:
        //Statement
    break;

    default:
        //Statement
    break;
}
```

```php
switch ($variable):
    case 1:
        //statement
        break;
    case 2:
        //statement
        break;
    default:
        //statement
endswitch;
```

CONDITIONALS (CẤU TRÚC ĐIỀU KHIỂN)

- Ví dụ

```
switch ($x)
{
  case 1: echo 'x is 1'; break;
  case 2: echo 'x is 2'; break;
  default: echo 'x is something else';
}

switch ($x):
  case 1:  echo 'x is 1'; break;
  case 2:  echo 'x is 2'; break;
  default: echo 'x is something else';
endswitch;
```


demo

PHẦN 2

TERNARY OPERATOR

- Ternary operator (Toán tử bậc ba): (?:)

Toán tử này có thể thay thế mệnh đề `if/else` đơn.

Toán tử cần 3 expressions. Nếu điều/cái đầu tiên đánh giá là `true`, thì biểu thức thứ 2 là giá trị trả về, và nếu là `false`, cái thứ 3 được trả về.

```
// Ternary operator expression
$y = ($x == 1) ? 1 : 2;

// Ternary operator statement
($x == 1) ? $y = 1 : $y = 2;
```

MATCH EXPRESSION

- PHP 8 đã thêm 1 câu lệnh điều kiện mới là ***match***.
- Match cung cấp một số cải tiến cho câu lệnh switch, cú pháp ngắn gọn hơn

```
switch ($x)
{
  case 1: $output = 'True' break;
  case 2: $output = 'False'; break;
  default: $output = 'Unknown';
}
```

```
$output = match($x) {
    1 => 'True',
    2 => 'False',
    default => 'Unknown'
};
echo $output;
```

MATCH EXPRESSION

- Trong cách trường hợp cần so sánh giá trị. Switch dùng ==, còn match dùng ===

```
$x = '1'; // string type
switch ($x)
{
  case 1: $output = 'integer'; break;
  case '1': $output = 'string';
}
echo $output; // "integer"

$output = match($x) {
  1 => 'integer',
  '1' => 'string',
};
echo $output; // "string"
```

MATCH EXPRESSION

- Các điều kiện xử lý giống nhau có thể được kết hợp trên cùng 1 dòng, cách nhau bởi dấu phẩy.
- Chú ý: match expression phải có điều kiện phù hợp hoặc có trường hợp default. Nếu không sẽ xảy ra lỗi.

```
$x = 4;
match($x) {
  1, 2, 3 => echo 'case 1, 2, or 3'
};
// Fatal error: Uncaught UnhandledMatchError
```

LOOP (CẤU TRÚC LẶP)

- While: chỉ chạy qua khối mã nếu điều kiện của nó là true. Lưu ý rằng điều kiện chỉ được kiểm tra khi bắt đầu mỗi lần lặp.

```
while (expr) {

    //statement

}
```

```
while (expr) :

    //statement

endwhile;
```

LOOP (CẤU TRÚC LẶP)

- Ví dụ

```
$i = 0;
$num = 50;
while( $i < 10){
    $num--;
    $i++;
}
echo ("Loop stopped at i = $i and num = $num" );
```

LOOP (CẤU TRÚC LẶP)

- Do…while: kiểm tra điều kiện sau khối mã. Do đó, do…while chạy qua khối mã ít nhất 1 lần..

```
do {
    //code to be executed;
}while (condition);
```

- Ví dụ

```
$i = 0;
$num = 50;
while( $i < 10){
    $num--;
    $i++;
}
echo ("Loop stopped at i = $i and num = $num" );
```

LOOP (CẤU TRÚC LẶP)

- For: Sử dụng lặp lại một khối mã và chạy nó đến khi điều kiện được đáp ứng
- For: được sử dụng nếu số lần lặp được biết trước
- Cú pháp:

```
for( counter initialization; condition; increments ) {
    //Statements
}
```

- hoặc

```
for (counter initialization; condition; increments):
    //statement
    //...
endfor;
```

Loop (Cấu trúc lặp)

- Ví dụ:

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

for($i = 0; $i < sizeof($a); $i++) {
  echo $a[$i]; // "123"
}
```

LOOP (CẤU TRÚC LẶP)

- `foreach`: sử dụng để duyệt qua các phần tử của mảng
- Cú pháp

```php
foreach ($array as $value) {
    //statements
}

foreach ($array as $key => $element) {
    //statements
}
```

- Hoặc

```php
foreach($array as $element):
  #do something
endforeach;
```

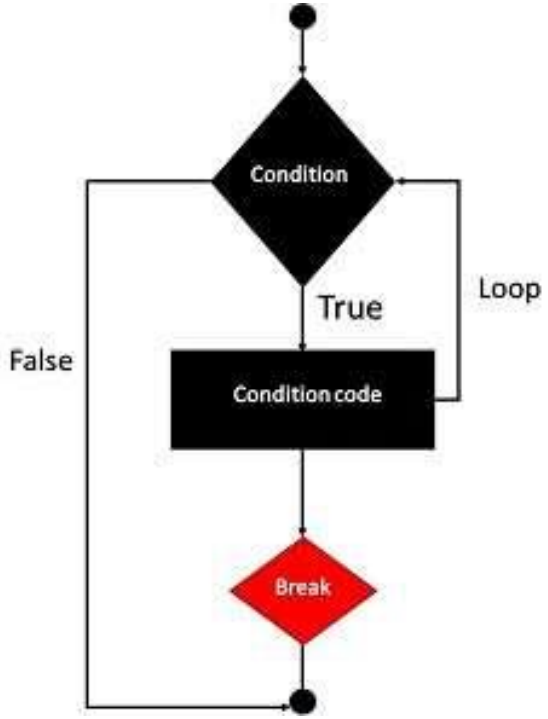
LOOP (CẤU TRÚC LẶP)

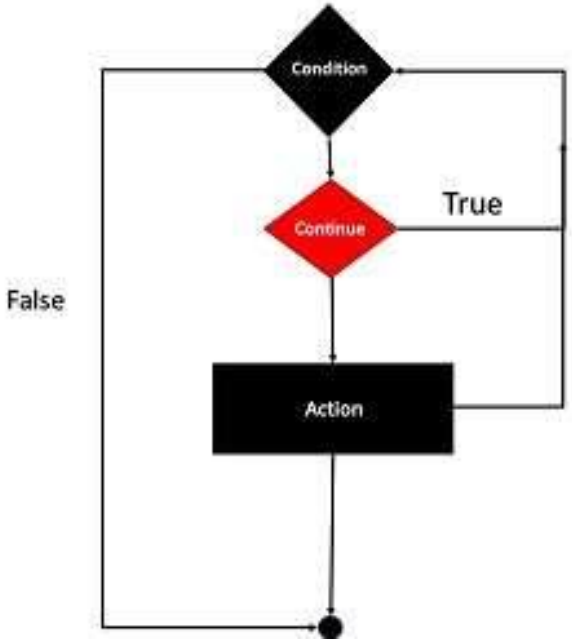
- `foreach`: sử dụng để duyệt qua các phần tử của mảng
- Ví dụ

```
$a = array('one' => 1, 'two' => 2);

foreach ($a as $k => $v) {
  echo "$k = $v "; // "one = 1 two = 2 "
}
```

BREAK, CONTINUE

- `break`: kết thúc quá trình thực thi vòng lặp
- `continue`: Tạm dung việc lặp lại hiện tại của vòng lặp nhưng nó không kết thúc vòng lặp.





BREAK, CONTINUE

Ví dụ

```php
$i = 0;
while( $i < 10)
{
    $i++;
    if( $i == 3 )break;
}
echo ("Loop stopped at i = $i" );
```

```
Loop stopped at i = 3
```

```php
$array = array( 1, 2, 3, 4, 5);
foreach( $array as $value )
{
  if( $value == 3 )continue;
  echo "Value is $value <br />";
}
```

```
Value is 1
Value is 2
Value is 4
Value is 5
```

GOTO

- Goto: thực hiện bước nhảy đến label được chỉ định.
- Label: là một tên, phía sau cód dấu hai chấm (:)
- Cú pháp

```
goto myLabel; // jump to label
myLabel: // label declaration
```

- Ví dụ

```
loop:
while (!$finished)
{
  // ...
  if ($try_again) goto loop; // restart loop
}
```


demo

SUMARRY

- ☑ Mảng (array)
- ☑ Hằng (constant)
- ☑ Cấu trúc điều khiển
- ☑ Cấu trúc lặp
- ☑ Break, continue, goto


thank
you!