Chương 1. Biến – Hằng Số - Toán Tử - Lệnh Điều Khiển

Trong tài liệu này, chúng ta sẽ khám phá về biến, hằng số, toán tử và các lệnh điều khiển.

Khái Niệm Biến Trong PHP

Biến trong PHP được sử dụng để lưu trữ dữ liệu, như số, chuỗi văn bản, hay các đối tượng phức tạp hơn. Tương tự chiếc hộp được dán nhãn, biến giúp bạn tổ chức và truy xuất thông tin trong chương trình.

Cú Pháp Khai Báo Biến

Tên biến luôn bắt đầu bằng ký tự $ và theo sau là tên của biến. Tên biến phân biệt chữ hoa và chữ thường, và chỉ có thể chứa các ký tự chữ cái, số và dấu gạch dưới (_). Tên biến không bắt đầu bằng số.

```
$ten_bien = giá_trị;
$hoTen = "Nguyễn Văn A";
$tuoi = 25;
$diem_trung_binh = 8.5;
```

Quy Tắc Đặt Tên Biến

- Luôn bắt đầu bằng ký tự $
- Chỉ chứa chữ cái, số và dấu gạch dưới
- Không được bắt đầu bằng số
- Phân biệt chữ hoa và chữ thường ($ten khác với $Ten)
- Nên đặt tên biến có ý nghĩa, liên quan đến dữ liệu mà nó lưu trữ

Biến có tính động, nghĩa là không cần phải khai báo kiểu dữ liệu trước khi sử dụng. PHP sẽ tự động chuyển đổi kiểu dữ liệu dựa trên giá trị được gán.

Phạm Vi Biến Trong PHP

Phạm vi của biến đề cập đến nơi mà biến có thể được truy cập trong mã nguồn. Trong PHP, có ba loại phạm vi biến chính: biến cục bộ (local), biến toàn cục (global), và biến tĩnh (static).

Biến Cục Bộ (Local)

Biến cục bộ được khai báo bên trong một hàm và chỉ có thể truy cập trong phạm vi của hàm đó. Khi hàm kết thúc, biến cục bộ sẽ bị hủy và giá trị của nó không còn được lưu trữ.

```
function tinhTong() {
    $a = 5; // Biến cục bộ
    $b = 10; // Biến cục bộ
    return $a + $b;
}
echo $a; // Lỗi! Không thể truy cập biến cục bộ từ bên ngoài hàm
```

Biến Toàn Cục (Global)

Biến toàn cục được khai báo bên ngoài tất cả các hàm và có thể được truy cập từ bất kỳ đâu trong tập tin PHP. Tuy nhiên, để sử dụng biến toàn cục bên trong một hàm, bạn cần sử dụng từ khóa **global** hoặc mảng $GLOBALS.

```
$x = 5; // Biến toàn cục

function suDungBienToanCuc() {
    global $x; // Khai báo sử dụng biến toàn cục
    echo $x;// Hiển thị giá trị 5
    // Hoặc sử dụng mảng $GLOBALS
    echo $GLOBALS['x']; // Cũng hiển thị giá trị 5
}
```

Biến Tĩnh (Static)

Biến tĩnh là biến cục bộ nhưng không bị hủy khi hàm kết thúc. Giá trị của biến tĩnh được giữ lại giữa các lần gọi hàm. Để khai báo biến tĩnh, sử dụng từ khóa **static**.

```
function demSoLanGoi() {
    static $dem = 0; // Biến tĩnh
    $dem++;
    echo "Hàm đã được gọi $dem lần";
}

demSoLanGoi(); // Hiển thị: Hàm đã được gọi 1 lần
demSoLanGoi(); // Hiển thị: Hàm đã được gọi 2 lần
```

Hằng Số Trong PHP

Hằng số là những giá trị không thể thay đổi sau khi được định nghĩa. Khác với biến, hằng số giữ nguyên giá trị của nó trong suốt thời gian thực thi chương trình. Hằng số thường được sử dụng để lưu trữ các giá trị không đổi như cấu hình hệ thống, các thông số kết nối cơ sở dữ liệu.

Sử dụng hàm define()

```
define("TEN_HANG", giá_trị);
define("PI", 3.14159);
define("WEBSITE_URL", "https://example.com");
define("MAX_USERS", 1000);
```

Hàm define() là cách truyền thống và linh hoạt nhất để định nghĩa hằng số. Hàm này có thể được sử dụng trong mọi ngữ cảnh, kể cả bên trong các điều kiện hoặc vòng lặp.

Sử dụng từ khóa const

```
const TEN_HANG = giá_trị;
const PI = 3.14159;
const WEBSITE_URL = "https://example.com";
const MAX_USERS = 1000;
```

Từ khóa const được giới thiệu từ PHP 5.3 và chỉ có thể được sử dụng ở cấp độ toàn cục hoặc trong khai báo lớp. Không thể sử dụng const bên trong các hàm, vòng lặp hoặc điều kiện if.

Đặc Điểm Của Hằng Số

- Không cần ký tự $ đứng trước tên hằng số
- Có thể truy cập từ bất kỳ đâu trong mã nguồn mà không cần từ khóa global
- Giá trị không thể thay đổi sau khi đã định nghĩa
- Thông thường, tên hằng số được viết hoàn toàn bằng chữ hoa để dễ phân biệt với biến
- Chỉ có thể chứa các giá trị vô hướng (scalar) như số, chuỗi, boolean hoặc NULL

Hằng Số Cấu Hình

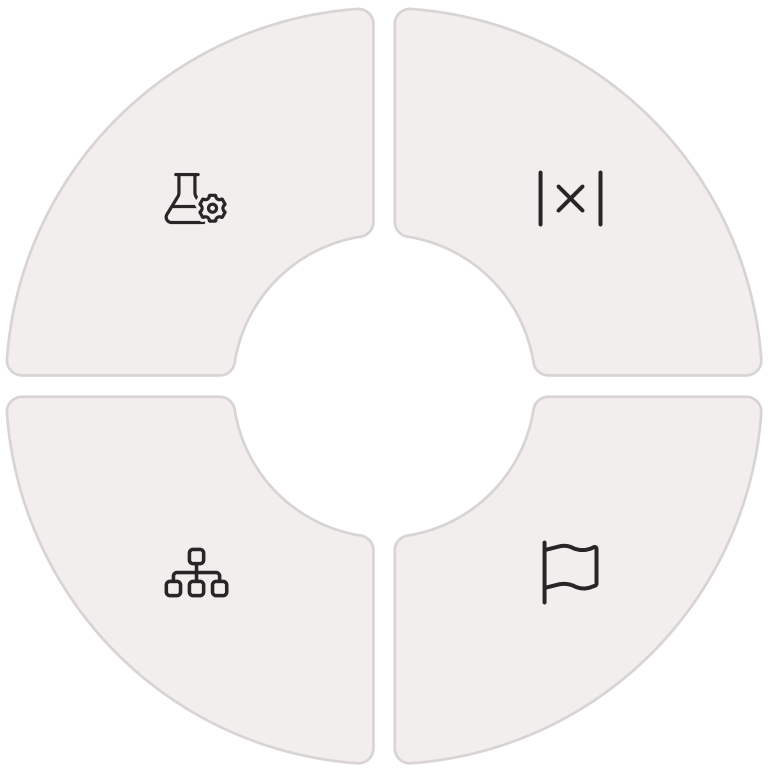
Lưu trữ thông tin cấu hình hệ thống như URL, thông số kết nối.

```
define("DB_HOST", "localhost");
define("DB_USER", "root");
```

Hằng Số Đường Dẫn

Lưu trữ đường dẫn thư mục.

```
define("ROOT_PATH", __DIR__);
define("UPLOAD_DIR", "uploads");
```



Hằng Số Toán Học

Các giá trị toán học không đổi.

```
define("PI", 3.14159);
define("G", 9.8);
```

Hằng Số Trạng Thái

Biểu thị các trạng thái hoặc mã lỗi.

```
define("STATUS_ACTIVE", 1);
define("ERROR_NOT_FOUND", 404);
```

PHP cũng cung cấp một số hằng số được định nghĩa sẵn như __FILE__, __LINE__, __DIR__, PHP_VERSION, PHP_OS. Những hằng số này cung cấp thông tin về môi trường PHP đang chạy.

Kiểu Dữ Liệu Trong PHP

Là ngôn ngữ có kiểu dữ liệu động, nghĩa là không cần phải khai báo kiểu dữ liệu khi tạo biến.

Kiểu Vô Hướng (Scalar Types)

- **Integer**: Số nguyên như 123, -456
- **Float/Double**: Số thực như 3.14, -2.718
- **Boolean**: Giá trị true hoặc false
- **String**: Chuỗi ký tự như "Hello, World!"

Kiểu Hợp Thành (Compound Types)

- **Array**: Mảng lưu trữ nhiều giá trị
- **Object**: Đối tượng trong lập trình hướng đối tượng

Kiểu Đặc Biệt (Special Types)

- **NULL**: Biểu thị biến không có giá trị
- **Resource**: Tham chiếu đến tài nguyên bên ngoài

Kiểu Integer (Số Nguyên)

Kiểu Integer đại diện cho các số nguyên không có phần thập phân. PHP có thể hiển thị số nguyên trong các hệ cơ số khác nhau: thập phân (cơ số 10), thập lục phân (cơ số 16), bát phân (cơ số 8) và nhị phân (cơ số 2).

```
$decimal = 123;      // số thập phân
$hexadecimal = 0x1A; // số thập lục phân (tương đương 26 trong hệ thập phân)
$octal = 0123;       // số bát phân (tương đương 83 trong hệ thập phân)
$binary = 0b1010;    // số nhị phân (tương đương 10 trong hệ thập phân)
```

Kiểu Float (Số Thực)

Kiểu Float (hoặc Double) biểu diễn các số có phần thập phân hoặc số được biểu diễn ở dạng khoa học.

```
$a = 1.234;
$b = 3e2;      // 3 * 10^2 = 300
$c = 4E-2;     // 4 * 10^(-2) = 0.04
```

Kiểu Boolean

Boolean đại diện cho một trong hai giá trị: TRUE hoặc FALSE. Trong ngữ cảnh của điều kiện, các giá trị sau được xem là FALSE:

- Chính giá trị boolean FALSE
- Số nguyên 0
- Số thực 0.0
- Chuỗi rỗng "" hoặc chuỗi "0"
- Mảng rỗng
- Đối tượng không có thuộc tính nào
- Giá trị NULL

Tất cả các giá trị khác được xem là TRUE.

Kiểu Dữ Liệu String và Array

Kiểu String (Chuỗi)

String là một chuỗi các ký tự, được đặt trong dấu nháy đơn (') hoặc dấu nháy kép ("). Sự khác biệt chính giữa hai loại dấu nháy này là cách xử lý các biến và ký tự đặc biệt (escape characters).

```
// Dấu nháy đơn
$str1 = 'Xin chào';
$name = 'Nam';
$str2 = 'Xin chào $name'; // Kết quả: "Xin chào $name"

// Dấu nháy kép
$str3 = "Xin chào";
$str4 = "Xin chào $name"; // Kết quả: "Xin chào Nam"

// Ký tự đặc biệt
$str5 = "Dòng 1\nDòng 2"; // \n tạo một dòng mới
```

Các hàm xử lý chuỗi mạnh mẽ như strlen(), strpos(), substr(), str_replace(), ...

```
$str = "Học PHP rất thú vị";
echo strlen($str);        // Độ dài chuỗi: 19
echo strpos($str, "PHP");  // Vị trí của "PHP": 4
echo substr($str, 4, 3);   // Cắt chuỗi: "PHP"
```

Kiểu Array (Mảng)

Array là một cấu trúc dữ liệu cho phép lưu trữ nhiều giá trị trong một biến duy nhất. PHP hỗ trợ cả mảng chỉ số (indexed arrays) và mảng kết hợp (associative arrays).

Mảng Chỉ Số

Mảng chỉ số sử dụng các chỉ số số nguyên làm khóa, bắt đầu từ 0.

```
// Khai báo mảng
$fruits = array("Táo", "Cam", "Chuối");
// Cú pháp ngắn gọn (PHP 5.4+)
$fruits = ["Táo", "Cam", "Chuối"];

// Truy cập phần tử
echo $fruits[0]; // Hiển thị: Táo
```

Mảng Kết Hợp

Mảng kết hợp sử dụng các chuỗi làm khóa thay vì chỉ số số nguyên.

```
// Khai báo mảng kết hợp
$person = array(
    "name" => "Nguyễn Văn A",
    "age" => 30,
    "city" => "Hà Nội"
);

// Truy cập phần tử
echo $person["name"]; // Hiển thị: Nguyễn Văn A
```

Mảng Đa Chiều

PHP cũng hỗ trợ mảng đa chiều, tức là mảng chứa các mảng khác.

```
// Mảng đa chiều
$employees = array(
    array("Nguyễn Văn A", "Giám đốc", 50000),
    array("Trần Thị B", "Nhân viên", 30000),
    array("Lê Văn C", "Kỹ thuật viên", 35000)
);

// Truy cập phần tử
echo $employees[0][0]; // Hiển thị: Nguyễn Văn A
echo $employees[1][2]; // Hiển thị: 30000
```

Sắp Xếp Mảng

PHP cung cấp nhiều hàm để sắp xếp mảng như sort(), rsort(), asort(), arsort(), ksort(), krsort().

```
$numbers = [5, 3, 8, 1];
sort($numbers); // Kết quả: [1, 3, 5, 8]
```

Lọc Mảng

Sử dụng hàm array_filter() để lọc các phần tử dựa trên một điều kiện.

```
$numbers = [1, 2, 3, 4, 5];
$even = array_filter($numbers, function($n) {
    return $n % 2 == 0;
}); // Kết quả: [2, 4]
```

Biến Đổi Mảng

Sử dụng hàm array_map() để áp dụng một hàm cho mỗi phần tử của mảng.

```
$numbers = [1, 2, 3, 4];
$squared = array_map(function($n) {
    return $n * $n;
}, $numbers); // Kết quả: [1, 4, 9, 16]
```

Ép Kiểu và Kiểm Tra Kiểu Dữ Liệu

PHP là ngôn ngữ có kiểu dữ liệu động, nhưng đôi khi bạn cần chuyển đổi một biến từ kiểu dữ liệu này sang kiểu dữ liệu khác, hoặc kiểm tra kiểu dữ liệu của một biến.

Ép Kiểu (Type Casting)

Ép kiểu là quá trình chuyển đổi một biến từ kiểu dữ liệu này sang kiểu dữ liệu khác. Trong PHP, có hai cách để thực hiện ép kiểu:

Ép kiểu ngầm định (Implicit)

PHP tự động chuyển đổi kiểu dữ liệu của biến dựa trên ngữ cảnh sử dụng. Ví dụ, khi bạn thực hiện phép cộng giữa một chuỗi số và một số nguyên, PHP sẽ tự động chuyển chuỗi thành số.

```
$str = "42";
// Chuỗi
$num = $str + 8;
// PHP tự động chuyển $str thành số
echo $num;
// Hiển thị: 50
```

Ép kiểu tường minh (Explicit)

Bằng cách sử dụng cú pháp ép kiểu hoặc các hàm chuyển đổi.

```
// Sử dụng cú pháp ép kiểu
$str = "42";
$num = (int)$str;    // Chuyển chuỗi thành số nguyên
$float = (float)$str; // Chuyển chuỗi thành số thực
$bool = (bool)$str;   // Chuyển chuỗi thành boolean

// Sử dụng hàm chuyển đổi
$num = intval($str);    // Chuyển thành số nguyên
$float = floatval($str); // Chuyển thành số thực
$bool = boolval($str);   // Chuyển thành boolean
```

Kiểm Tra Kiểu Dữ Liệu

PHP cung cấp nhiều hàm để kiểm tra kiểu dữ liệu của một biến:

Hàm	Mô tả	Ví dụ
is_int()	Kiểm tra xem biến có phải là số nguyên không	is_int(42) // true
is_float()	Kiểm tra xem biến có phải là số thực không	is_float(3.14) // true
is_string()	Kiểm tra xem biến có phải là chuỗi không	is_string("PHP") // true
is_bool()	Kiểm tra xem biến có phải là boolean không	is_bool(true) // true
is_array()	Kiểm tra xem biến có phải là mảng không	is_array([1,2,3]) // true
is_null()	Kiểm tra xem biến có phải là NULL không	is_null(null) // true
is_object()	Kiểm tra xem biến có phải là đối tượng không	is_object(new stdClass()) // true
gettype()	Trả về kiểu dữ liệu của biến dưới dạng chuỗi	gettype(42) // "integer"
var_dump()	Hiển thị thông tin chi tiết về biến	var_dump(42) // int(42)

Ví dụ Thực Tế

```
// Kiểm tra và ép kiểu dữ liệu nhập vào từ form
$age = $_POST['age'];

if (!is_numeric($age)) {
    echo "Tuổi phải là một số!";
} else {
    $age = (int)$age; // Ép kiểu thành số nguyên

    if ($age < 18) {
        echo "Bạn chưa đủ tuổi!";
    } else {
        echo "Xin chào, bạn đã $age tuổi!";
    }
}
```

Nhận Dữ Liệu

Dữ liệu từ form, database, API thường có kiểu dữ liệu không xác định

Xuất Kết Quả

Hiển thị hoặc trả về kết quả cho người dùng

Kiểm Tra Kiểu

Sử dụng các hàm is_* để xác định kiểu dữ liệu hiện tại

Ép Kiểu

Chuyển đổi dữ liệu sang kiểu thích hợp cho xử lý

Xử Lý Dữ Liệu

Thực hiện các phép tính, so sánh hoặc lưu trữ dữ liệu

Toán Tử Trong PHP

Toán tử trong PHP là các ký hiệu đặc biệt được sử dụng để thực hiện các phép tính hoặc thao tác trên các biến và giá trị.

Toán Tử Số Học (Arithmetic Operators)

Toán tử số học được sử dụng để thực hiện các phép tính toán học cơ bản.

Toán tử	Tên	Ví dụ	Kết quả
+	Cộng	$a + $b	Tổng của $a và $b
-	Trừ	$a - $b	Hiệu của $a và $b
*	Nhân	$a * $b	Tích của $a và $b
/	Chia	$a / $b	Thương của $a và $b
%	Chia lấy dư	$a % $b	Phần dư khi chia $a cho $b
**	Lũy thừa	$a ** $b	$a lũy thừa $b

```
$a = 10;
$b = 3;

echo $a + $b;  // 13
echo $a - $b;  // 7
echo $a * $b;  // 30
echo $a / $b;  // 3.3333...
echo $a % $b;  // 1 (phần dư của 10 ÷ 3)
echo $a ** $b; // 1000 (10^3)
```

Toán Tử Gán (Assignment Operators)

Toán tử gán được sử dụng để gán giá trị cho biến.

Toán tử gán cơ bản

```
$a = 5; // Gán giá trị 5 cho biến $a
```

Toán tử gán kết hợp

```
$a += 3;  // Tương đương với $a = $a + 3
$a -= 2;  // Tương đương với $a = $a - 2
$a *= 4;  // Tương đương với $a = $a * 4
$a /= 2;  // Tương đương với $a = $a / 2
$a %= 3;  // Tương đương với $a = $a % 3
```

Toán Tử So Sánh (Comparison Operators)

Toán tử so sánh được sử dụng để so sánh hai giá trị và trả về giá trị boolean (true hoặc false).

Toán tử	Tên	Ví dụ	Kết quả
==	Bằng	$a == $b	TRUE nếu $a bằng $b sau khi chuyển đổi kiểu dữ liệu
===	Bằng nghiêm ngặt	$a === $b	TRUE nếu $a bằng $b và cùng kiểu dữ liệu
!=	Khác	$a != $b	TRUE nếu $a khác $b sau khi chuyển đổi kiểu dữ liệu
!==	Khác nghiêm ngặt	$a !== $b	TRUE nếu $a khác $b hoặc khác kiểu dữ liệu
>	Lớn hơn	$a > $b	TRUE nếu $a lớn hơn $b
<	Nhỏ hơn	$a < $b	TRUE nếu $a nhỏ hơn $b
>=	Lớn hơn hoặc bằng	$a >= $b	TRUE nếu $a lớn hơn hoặc bằng $b
<=	Nhỏ hơn hoặc bằng	$a <= $b	TRUE nếu $a nhỏ hơn hoặc bằng $b
<=>	Spaceship	$a <=> $b	Trả về -1 nếu $a < $b, 0 nếu $a == $b, 1 nếu $a > $b

Toán Tử Tăng/Giảm (Increment/Decrement Operators)

Toán tử tăng/giảm được sử dụng để tăng hoặc giảm giá trị của biến đi một đơn vị.

```
$a = 5;
$a++; // Tăng $a lên 1 sau khi sử dụng giá trị hiện tại (post-increment)
++$a; // Tăng $a lên 1 trước khi sử dụng giá trị (pre-increment)
$a--; // Giảm $a đi 1 sau khi sử dụng giá trị hiện tại (post-decrement)
--$a; // Giảm $a đi 1 trước khi sử dụng giá trị (pre-decrement)
```

Toán Tử Logic và Toán Tử Chuỗi

Toán Tử Logic (Logical Operators)

Toán tử logic được sử dụng để kết hợp các điều kiện và trả về giá trị boolean (true hoặc false). Chúng rất hữu ích trong các câu lệnh điều kiện như if, while, v.v.

Toán tử	Tên	Ví dụ	Kết quả
&&	AND (và)	$a && $b	TRUE nếu cả $a và $b đều TRUE
||	OR (hoặc)	$a || $b	TRUE nếu $a hoặc $b là TRUE
!	NOT (phủ định)	!$a	TRUE nếu $a là FALSE, FALSE nếu $a là TRUE
and	AND (và)	$a and $b	Giống &&, nhưng có độ ưu tiên thấp hơn
or	OR (hoặc)	$a or $b	Giống ||, nhưng có độ ưu tiên thấp hơn
xor	XOR (hoặc loại trừ)	$a xor $b	TRUE nếu $a hoặc $b là TRUE, nhưng không phải cả hai

```
$a = true;
$b = false;

var_dump($a && $b); // bool(false)
var_dump($a || $b); // bool(true)
var_dump(!$a);      // bool(false)
var_dump($a xor $b); // bool(true)
```

Chú ý sự khác biệt về độ ưu tiên giữa các toán tử && và and, || và or:

```
$a = false;
$b = true;
$c = false;

// Sử dụng &&
$result1 = $a && $b || $c; // ($a && $b) || $c = (false && true) || false = false || false = false

// Sử dụng and
$result2 = $a and $b || $c; // $a and ($b || $c) = false and (true || false) = false and true = false
```

Toán Tử Chuỗi (String Operators)

PHP cung cấp hai toán tử chính để làm việc với chuỗi: toán tử nối chuỗi (.) và toán tử gán nối chuỗi (.=).

Toán Tử Nối Chuỗi (.)

Toán tử này sử dụng để nối hai chuỗi lại với nhau.

```
$firstName = "Nguyễn";
$lastName = "Văn A";
$fullName = $firstName . " " . $lastName;
echo $fullName;
// Hiển thị: Nguyễn Văn A
```

Toán Tử Gán Nối Chuỗi (.=)

Toán tử này nối chuỗi bên phải vào chuỗi bên trái và gán kết quả cho chuỗi bên trái.

```
$greeting = "Xin chào ";
$greeting .= "các bạn!";
echo $greeting; // Hiển thị: Xin chào các bạn!
```

Thứ Tự Ưu Tiên Của Toán Tử

Đánh giá các biểu thức theo thứ tự ưu tiên của toán tử. Toán tử có độ ưu tiên cao hơn sẽ được thực hiện trước. Dưới đây là thứ tự ưu tiên từ cao đến thấp của một số toán tử phổ biến:

1. () (nhóm biểu thức)
2. ++ -- (tăng giảm)
3. ! (phủ định)
4. * / % (nhân, chia, chia lấy dư)
5. + - (cộng, trừ)
6. < <= > >= (so sánh)
7. == != === !== <> (bằng, khác)
8. && (AND logic)
9. || (OR logic)
10. = += -= *= /= %= .= (gán)
11. and (AND logic với độ ưu tiên thấp)
12. xor (XOR logic)
13. or (OR logic với độ ưu tiên thấp)

Nếu không chắc chắn về thứ tự ưu tiên, thì nên sử dụng dấu ngoặc đơn () để nhóm các biểu thức và làm rõ thứ tự thực hiện.

Toán Tử Đặc Biệt Trong PHP

Ngoài các toán tử cơ bản như số học, logic, và so sánh, PHP còn cung cấp một số toán tử đặc biệt với các tính năng độc đáo. Những toán tử này giúp mã của bạn ngắn gọn và hiệu quả hơn.

Toán Tử Ba Ngôi (Ternary Operator)

Toán tử ba ngôi là một cách viết ngắn gọn của câu lệnh if-else.

Cú pháp: `điều_kiện ? giá_trị_nếu_đúng : giá_trị_nếu_sai`.

```
// Thay vì viết:
if ($age >= 18) {
    $status = "Người lớn";
} else {
    $status = "Trẻ em";
}

// Bạn có thể viết ngắn gọn hơn:
$status = ($age >= 18) ? "Người lớn" : "Trẻ em";
```

PHP cũng hỗ trợ toán tử ba ngôi lồng nhau, nhưng nên tránh sử dụng quá nhiều vì sẽ làm mã khó đọc:

```
$result = ($age < 13) ? "Trẻ em" : (($age < 18) ? "Thanh thiếu niên" : "Người lớn");
```

Toán Tử Null Coalescing (??)

Toán tử này được giới thiệu từ PHP 7 và là một cách ngắn gọn để kiểm tra xem một biến có tồn tại và không phải null không.

Cú pháp: `biểu_thức1 ?? biểu_thức2`, trả về biểu_thức1 nếu nó tồn tại và khác null, ngược lại trả về biểu_thức2.

```
// Thay vì viết:
$username = isset($_GET['user']) ? $_GET['user'] : 'Khách';

// Bạn có thể viết ngắn gọn hơn:
$username = $_GET['user'] ?? 'Khách';
```

Toán tử này đặc biệt hữu ích khi làm việc với dữ liệu từ form, API, hoặc cơ sở dữ liệu, nơi cần xử lý các giá trị có thể không tồn tại.

Toán Tử Spread (...)

Toán tử spread được giới thiệu từ PHP 7.4 và cho phép bạn "trải" các phần tử của một mảng vào một mảng khác hoặc vào danh sách tham số của một hàm.

```
// Kết hợp mảng
$fruits1 = ['táo', 'cam'];
$fruits2 = ['chuối', 'nho'];
$allFruits = [...$fruits1, ...$fruits2]; // ['táo', 'cam', 'chuối', 'nho']

// Truyền mảng làm tham số cho hàm
function sum($a, $b, $c) {
    return $a + $b + $c;
}
$numbers = [1, 2, 3];
echo sum(...$numbers); // 6
```

Toán Tử Gán Null Coalescing (??=)

Toán tử này được giới thiệu từ PHP 7.4 và gán giá trị cho biến chỉ khi biến đó là null.

Có cú pháp: `$biến ??= giá_trị`.

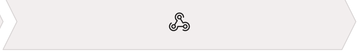
```
// Thay vì viết:
if ($username === null) {
    $username = 'Khách';
}

// Hoặc:
$username = $username ?? 'Khách';

// Bạn có thể viết ngắn gọn hơn:
$username ??= 'Khách';
```

Kiểm Tra Điều Kiện

Sử dụng toán tử ba ngôi (? :) cho các điều kiện đơn giản thay vì if-else.



Xử Lý Giá Trị Null

Sử dụng toán tử null coalescing (??) để cung cấp giá trị mặc định khi biến là null.

Kết Hợp Mảng

Sử dụng toán tử spread (...) để kết hợp các mảng hoặc truyền mảng làm tham số.

Gán Giá Trị Mặc Định

Sử dụng toán tử gán null coalescing (??=) để gán giá trị chỉ khi biến là null.

Ví dụ Thực Tế

Dưới đây là một ví dụ kết hợp các toán tử đặc biệt để xử lý dữ liệu đầu vào từ người dùng:

```
// Lấy thông tin người dùng từ form hoặc sử dụng giá trị mặc định
$username = $_POST['username'] ?? 'Khách';
$age = (int)($_POST['age'] ?? 0); // Chuyển đổi sang số nguyên
$preferences = $_POST['preferences'] ?? [];

// Mặc định bật thông báo email nếu người dùng chưa chọn
$notifications = $_POST['notifications'] ?? [];
$notifications['email'] ??= true;

// Kết hợp sở thích mặc định với sở thích của người dùng
$defaultPreferences = ['theme' => 'light', 'language' => 'vi'];
$userPreferences = [...$defaultPreferences, ...$preferences];

// Hiển thị thông báo phù hợp với độ tuổi
$message = ($age < 18)
    ? "Xin chào bạn trẻ $username!"
    : (($age < 60) ? "Chào mừng $username!" : "Kính chào $username!");

echo $message;
```

Các toán tử đặc biệt này giúp mã của bạn ngắn gọn và dễ đọc hơn, đồng thời giảm thiểu lỗi khi xử lý các trường hợp đặc biệt như giá trị null hoặc không tồn tại.

Cấu Trúc Điều Kiện Trong PHP

Cấu trúc điều kiện là một phần không thể thiếu trong bất kỳ ngôn ngữ lập trình nào, cho phép chương trình thực hiện các hành động khác nhau dựa trên các điều kiện khác nhau.

Câu lệnh if cơ bản nhất cho phép thực thi một khối mã nếu điều kiện được đánh giá là TRUE.

```
if (điều_kiện) {
    // Mã được thực thi nếu điều kiện là TRUE
}
```

Ví dụ:

```
$tuoi = 25;

if ($tuoi >= 18) {
    echo "Bạn đã đủ tuổi trưởng thành.";
}
```

Câu Lệnh if-else

Câu lệnh if-else cho phép thực thi một khối mã nếu điều kiện là TRUE và một khối mã khác nếu điều kiện là FALSE.

```
if (điều_kiện) {
    // Mã được thực thi nếu điều kiện là TRUE
} else {
    // Mã được thực thi nếu điều kiện là FALSE
}
```

Ví dụ:

```
$diem = 75;

if ($diem >= 50) {
    echo "Bạn đã đậu.";
} else {
    echo "Bạn đã trượt.";
}
```

Câu Lệnh if-elseif-else

Câu lệnh if-elseif-else cho phép kiểm tra nhiều điều kiện và thực thi các khối mã tương ứng.

```
if (điều_kiện1) {
    // Mã được thực thi nếu điều_kiện1 là TRUE
} elseif (điều_kiện2) {
    // Mã được thực thi nếu điều_kiện1 là FALSE và điều_kiện2 là TRUE
} else {
    // Mã được thực thi nếu tất cả các điều kiện trên là FALSE
}
```

Ví dụ:

```
$diem = 85;

if ($diem >= 90) {
    echo "Xuất sắc";
} elseif ($diem >= 80) {
    echo "Giỏi";
} elseif ($diem >= 70) {
    echo "Khá";
} elseif ($diem >= 60) {
    echo "Trung bình";
} else {
    echo "Yếu";
}
```

Câu Lệnh switch

Câu lệnh switch là một cách khác để kiểm tra nhiều điều kiện. Hữu ích khi bạn cần so sánh một biến với nhiều giá trị khác nhau.

```
switch (biểu_thức) {
    case giá_trị1:
        // Mã được thực thi nếu biểu_thức == giá_trị1
        break;
    case giá_trị2:
        // Mã được thực thi nếu biểu_thức == giá_trị2
        break;
    ...
    default:
        // Mã được thực thi nếu biểu_thức không khớp với bất kỳ giá trị nào
}
```

Ví dụ:

```
$ngay = 3;

switch ($ngay) {
    case 1:
        echo "Thứ Hai";
        break;
    case 2:
        echo "Thứ Ba";
        break;
    case 3:
        echo "Thứ Tư";
        break;
    case 4:
        echo "Thứ Năm";
        break;
    case 5:
        echo "Thứ Sáu";
        break;
    case 6:
        echo "Thứ Bảy";
        break;
    case 7:
        echo "Chủ Nhật";
        break;
    default:
        echo "Ngày không hợp lệ";
}
```

Lưu ý: Từ khóa `break` rất quan trọng trong câu lệnh switch. Nếu quên break, PHP sẽ tiếp tục thực thi mã trong các case tiếp theo, ngay cả khi điều kiện không khớp.

Cấu Trúc Điều Kiện Nâng Cao

Cú Pháp Rút Gọn Của if-else

PHP cho phép bạn viết các câu lệnh if-else một cách ngắn gọn nếu trong mỗi khối chỉ có một câu lệnh. Bạn có thể bỏ qua dấu ngoặc nhọn {} trong trường hợp này.

```
// Cú pháp đầy đủ
if ($tuoi >= 18) {
    echo "Bạn đã đủ tuổi.";
} else {
    echo "Bạn chưa đủ tuổi.";
}

// Cú pháp rút gọn
if ($tuoi >= 18)
    echo "Bạn đã đủ tuổi.";
else
    echo "Bạn chưa đủ tuổi.";
```

Tuy nhiên, cách viết này có thể dẫn đến lỗi khi thêm câu lệnh vào sau này.

Câu Lệnh match (PHP 8+)

Từ PHP 8.0, một cấu trúc điều kiện mới được giới thiệu là match. Nó tương tự như switch nhưng có một số khác biệt quan trọng:

- match sử dụng so sánh nghiêm ngặt (===) thay vì so sánh lỏng lẻo (==)
- match trả về một giá trị, có thể gán cho biến
- match không yêu cầu từ khóa break
- match có thể xử lý nhiều giá trị cho một trường hợp

```
$ngay = 3;

$tenNgay = match ($ngay) {
    1 => "Thứ Hai",
    2 => "Thứ Ba",
    3 => "Thứ Tư",
    4 => "Thứ Năm",
    5 => "Thứ Sáu",
    6, 7 => "Cuối tuần",
    default => "Ngày không hợp lệ",
};

echo $tenNgay; // Hiển thị: Thứ Tư
```

Toán Tử Điều Kiện Trong Chuỗi

Nhúng các biểu thức điều kiện trực tiếp vào chuỗi sử dụng cú pháp `{biểu_thức}` khi chuỗi được đặt trong dấu nháy kép.

```
$tuoi = 25;
echo "Bạn {$tuoi < 18 ? 'chưa' : 'đã'} đủ trưởng thành.";
// Hiển thị: Bạn trưởng thành.
```

Cách này rất hữu ích khi bạn muốn hiển thị thông điệp có điều kiện mà không cần chia nhỏ chuỗi.

Kiểm Tra Nhiều Điều Kiện

Khi cần kiểm tra một biến với nhiều giá trị, thay vì viết nhiều điều kiện OR (||), có thể sử dụng hàm `in_array()`:

```
$trangThai = "đang xử lý";

// Thay vì viết:
if ($trangThai == "đang chờ" || $trangThai == "đang xử lý" || $trangThai == "đang giao") {
    echo "Đơn hàng đang được xử lý.";
}

// Viết ngắn gọn hơn:
if (in_array($trangThai, ["đang chờ", "đang xử lý", "đang giao"])) {
    echo "Đơn hàng đang được xử lý.";
}
```

Null Coalescing Operator Trong Điều Kiện

Toán tử null coalescing (??) rất hữu ích trong các cấu trúc điều kiện khi bạn cần kiểm tra xem một biến có tồn tại và không phải null không:

```
// Thay vì viết:
if (isset($user) && $user !== null) {
    $userName = $user;
} else {
    $userName = "Khách";
}

// Bạn có thể viết ngắn gọn hơn:
$userName = $user ?? "Khách";
```

1. **Kiểm Tra Điều Kiện**
   Sử dụng if-else hoặc switch để đánh giá điều kiện và quyết định hành động.
2. **Kiểm Tra Nhiều Điều Kiện**
   Sử dụng if-elseif-else hoặc switch khi cần kiểm tra nhiều điều kiện khác nhau.
3. **Kiểm Tra Một Biến Với Nhiều Giá Trị**
   Sử dụng in_array() hoặc match để kiểm tra một biến với nhiều giá trị có thể.
4. **Xử Lý Giá Trị Null Hoặc Không Tồn Tại**
   Sử dụng toán tử null coalescing (??) để cung cấp giá trị mặc định khi biến là null hoặc không tồn tại.
5. **Trả Về Giá Trị Có Điều Kiện**
   Sử dụng toán tử ba ngôi hoặc match để trả về giá trị dựa trên điều kiện.

Cấu Trúc Vòng Lặp Trong PHP

Vòng lặp là một cấu trúc lập trình cho phép bạn thực thi một khối mã nhiều lần.

Vòng Lặp while

Vòng lặp while thực thi một khối mã lặp đi lặp lại miễn là điều kiện được đánh giá là TRUE. Điều kiện được kiểm tra trước mỗi lần lặp.

```
while (điều_kiện) {
    // Mã được thực thi lặp đi lặp lại miễn là điều kiện là TRUE
}
```

Ví dụ:

```
$i = 1;
while ($i <= 5) {
    echo "Số $i
";
    $i++;
}
```

Kết quả:

```
Số 1
Số 2
Số 3
Số 4
Số 5
```

Vòng Lặp do-while

Vòng lặp do-while tương tự như vòng lặp while, nhưng nó kiểm tra điều kiện sau khi thực thi khối mã. Điều này đảm bảo rằng khối mã được thực thi ít nhất một lần, ngay cả khi điều kiện ban đầu là FALSE.

```
do {
    // Mã được thực thi ít nhất một lần
} while (điều_kiện);
```

Ví dụ:

```
$i = 1;
do {
    echo "Số $i
";
    $i++;
} while ($i <= 5);
```

Kết quả cũng giống như ví dụ while ở trên.

Vòng Lặp for

Vòng lặp for được sử dụng khi bạn biết trước số lần lặp. Nó bao gồm ba phần: khởi tạo, điều kiện, và tăng/giảm biến đếm.

```
for (khởi_tạo; điều_kiện; tăng/giảm) {
    // Mã được thực thi lặp đi lặp lại
}
```

Ví dụ:

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

Vòng Lặp foreach

Vòng lặp foreach được thiết kế đặc biệt để làm việc với mảng. Nó lặp qua mỗi phần tử của mảng và gán giá trị của phần tử hiện tại cho một biến.

Cú pháp cơ bản

```
foreach ($mang as $gia_tri) {
    // Mã được thực thi cho mỗi phần tử của mảng
    // $gia_tri chứa giá trị của phần tử hiện tại
}
```

Ví dụ:

```
$colors = ["red", "green", "blue"];
foreach ($colors as $color) {
    echo "Màu: $color
";
}
```

Kết quả:

```
Màu: red
Màu: green
Màu: blue
```

Làm việc với khóa và giá trị

```
foreach ($mang as $khoa => $gia_tri) {
    // Mã được thực thi cho mỗi phần tử của mảng
    // $khoa chứa khóa của phần tử hiện tại
    // $gia_tri chứa giá trị của phần tử hiện tại
}
```

Ví dụ:

```
$person = [
    "name" => "Nguyễn Văn A",
    "age" => 30,
    "city" => "Hà Nội"
];
foreach ($person as $key => $value) {
    echo "$key: $value
";
}
```

Kết quả:

```
name: Nguyễn Văn A
age: 30
city: Hà Nội
```

while

Sử dụng khi bạn không biết trước số lần lặp và muốn kiểm tra điều kiện trước khi thực thi khối mã.

do-while

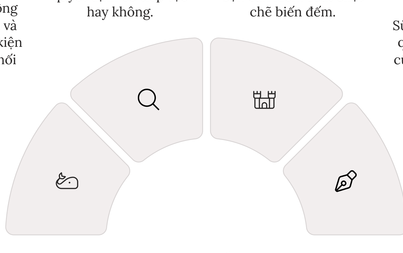
Sử dụng khi bạn muốn khối mã được thực thi ít nhất một lần, sau đó kiểm tra điều kiện để quyết định có tiếp tục hay không.

for

Sử dụng khi bạn biết trước số lần lặp hoặc khi bạn cần kiểm soát chặt chẽ biến đếm.

foreach

Sử dụng khi bạn cần lặp qua tất cả các phần tử của một mảng, đặc biệt là mảng kết hợp.



Điều Khiển Vòng Lặp và Xử Lý Lỗi

Lệnh Điều Khiển Vòng Lặp

PHP cung cấp các lệnh đặc biệt để điều khiển luồng thực thi trong vòng lặp. Hai lệnh phổ biến nhất là break và continue.

Lệnh break

Lệnh break được sử dụng để thoát khỏi vòng lặp hiện tại. Khi gặp lệnh break, PHP sẽ kết thúc vòng lặp ngay lập tức và tiếp tục thực thi mã sau vòng lặp.

```
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        break; // Thoát khỏi vòng lặp khi $i =
5
    }
    echo "Số $i
";
}
// Kết quả: Số 1, Số 2, Số 3, Số 4
```

Lệnh continue

Lệnh continue được sử dụng để bỏ qua phần còn lại của vòng lặp hiện tại và tiếp tục với lần lặp tiếp theo. Khi gặp lệnh continue, PHP sẽ bỏ qua các câu lệnh còn lại trong vòng lặp và bắt đầu lần lặp mới.

```
for ($i = 1; $i <= 5; $i++) {
    if ($i == 3) {
        continue; // Bỏ qua lần lặp khi $i = 3
    }
    echo "Số $i
";
}
// Kết quả: Số 1, Số 2, Số 4, Số 5
```

Xử Lý Lỗi Trong PHP

Cấu Trúc try-catch

Cấu trúc try-catch cho phép bắt và xử lý các ngoại lệ (exceptions) xảy ra trong khối mã try. Giúp chương trình không bị dừng đột ngột khi gặp lỗi.

```
try {
    // Mã có thể gây ra ngoại lệ
    $result = 10 / 0; // Gây ra lỗi chia cho 0
} catch (Exception $e) {
    // Mã xử lý ngoại lệ
    echo "Đã xảy ra lỗi: " . $e->getMessage();
}
```

Từ PHP 7.0 trở đi, bạn có thể bắt nhiều loại ngoại lệ khác nhau:

```
try {
    // Mã có thể gây ra ngoại lệ
    if (!file_exists('file.txt')) {
        throw new Exception("File không tồn tại");
    }
    // Mã khác có thể gây ra ngoại lệ khác
} catch (FileException $e) {
    // Xử lý ngoại lệ liên quan đến file
    echo "Lỗi file: " . $e->getMessage();
} catch (Exception $e) {
    // Xử lý tất cả các ngoại lệ khác
    echo "Lỗi chung: " . $e->getMessage();
} finally {
    // Mã luôn được thực thi, bất kể có ngoại lệ hay không
    echo "Đã hoàn thành xử lý.";
}
```

Hàm die() và exit()

Các hàm die() và exit() được sử dụng để dừng thực thi script ngay lập tức. Chúng thường được sử dụng trong quá trình phát triển hoặc khi gặp lỗi nghiêm trọng.

```
$file = fopen("test.txt", "r");
if (!$file) {
    die("Không thể mở file.");
}
// Mã xử lý file ở đây
```

Bài Tập Thực Hành

I. Toán Tử và Biểu Thức

Bài 1: Toán Tử Số Học

Viết chương trình cho công ty xây dựng, sử dụng các toán tử số học (+, -, *, /, %, **) để tính diện tích và thể tích của hình học cơ bản (hình tròn, hình vuông, hình chữ nhật). Kết quả sẽ được sử dụng để ước tính vật liệu xây dựng và chi phí thi công.

Bài 2: Toán Tử So Sánh

Tạo ứng dụng cho bộ phận mua hàng, so sánh giá của hai nhà cung cấp khác nhau và hiển thị kết quả phân tích sử dụng tất cả các toán tử so sánh (==, ===, !=, !==, >, <, >=, <=). Ứng dụng sẽ giúp đưa ra quyết định mua hàng tối ưu.

Bài 3: Toán Tử Logic

Xây dựng hệ thống xác định điểm học phần cho trường đại học, dựa trên nhiều điều kiện kết hợp (chuyên cần 10%, giữa kỳ 30%, cuối kỳ 60%) sử dụng toán tử &&, ||, và !. Hệ thống cần xử lý các trường hợp điểm cộng, điểm trừ và các điều kiện đặc biệt như vắng thi.

II. Cấu Trúc Điều Kiện

Bài 1: If-Else

Viết chương trình cho công ty du lịch, kiểm tra năm nhuận với điều kiện lồng nhau, xử lý các trường hợp ngoại lệ trong lịch. Ứng dụng sẽ giúp tính toán chính xác thời gian của các tour du lịch và kỳ nghỉ lễ trong nhiều năm.

Bài 2: Switch-Case

Tạo ứng dụng tính lương nhân viên cho bộ phận nhân sự, với nhiều mức theo chức vụ (nhân viên, quản lý, giám đốc), thâm niên (dưới 1 năm, 1-3 năm, trên 3 năm) và hiệu suất làm việc (A, B, C) sử dụng switch-case. Hệ thống cần tính cả phụ cấp và thuế thu nhập.

Bài 3: Toán Tử Ba Ngôi

Xây dựng hệ thống hiển thị thông báo tình trạng đơn hàng cho website thương mại điện tử, sử dụng toán tử ba ngôi lồng nhau cho nhiều tình huống khác nhau (đơn hàng mới, đang xử lý, đang giao, đã giao, bị hủy, hoàn trả). Hệ thống cần hiển thị màu sắc và biểu tượng phù hợp với từng trạng thái.

III. Cấu Trúc Vòng Lặp

Bài 1: For Loop

Tạo chương trình cho ứng dụng học toán dành cho học sinh tiểu học, in ra bảng cửu chương hoàn chỉnh từ 1 đến 10 với định dạng bảng HTML, sử dụng vòng lặp for lồng nhau. Bảng cần có màu sắc và định dạng thân thiện với người dùng, với khả năng ẩn/hiện kết quả để kiểm tra.

Bài 2: While và Do-While

Viết ứng dụng mô phỏng trò chơi đoán số cho website giải trí, với số lần đoán giới hạn (tối đa 10 lần), sử dụng cả while và do-while để so sánh sự khác biệt. Người chơi sẽ nhận được gợi ý (cao hơn/thấp hơn) sau mỗi lần đoán và điểm thưởng dựa trên số lần đoán.

Bài 3: Foreach với Array

Xây dựng chương trình quản lý sinh viên cho phòng đào tạo, với mảng đa chiều lưu trữ thông tin cá nhân (mã SV, họ tên, ngày sinh), điểm các môn học (tối thiểu 5 môn), hiển thị thông tin và tính điểm trung bình, xếp loại sử dụng foreach. Hệ thống cần có chức năng lọc sinh viên theo điểm và xuất báo cáo.

IV. Xử Lý Lỗi

Bài 1: Try-Catch Cơ Bản

Viết chương trình xử lý lỗi cho ứng dụng ngân hàng, bao gồm xử lý lỗi khi chia cho số 0 (tính toán lãi suất), đọc file không tồn tại (báo cáo giao dịch), và kết nối cơ sở dữ liệu thất bại (kiểm tra số dư). Chương trình cần ghi nhật ký lỗi và hiển thị thông báo thân thiện cho người dùng.

V. Dự Án Thực Hành Tổng Hợp

Bài 1: Ứng Dụng Quản Lý Thư Viện

Xây dựng ứng dụng quản lý sách cho thư viện trường học, sử dụng mảng đa chiều lưu trữ thông tin sách (mã sách, tên, tác giả, năm xuất bản, thể loại, vị trí), cấu trúc điều kiện để kiểm tra tình trạng sách (có sẵn, đang mượn, đang bảo trì) và vòng lặp để hiển thị danh sách sách theo thể loại. Hệ thống cần có chức năng mượn/trả và tính phí trễ hạn.

Bài 2: Hệ Thống Tính Điểm Học Sinh

Tạo chương trình tính điểm trung bình và xếp loại học sinh cho trường THPT, sử dụng mảng để lưu trữ điểm các môn học (Toán, Văn, Anh, Lý, Hóa, Sinh, Sử, Địa), hệ số từng môn, cấu trúc điều kiện if-else để xác định xếp loại (Giỏi, Khá, Trung bình, Yếu) và vòng lặp để xử lý nhiều học sinh. Chương trình cần có chức năng phân tích thống kê và xuất bảng điểm theo lớp.

Bài 3: Giỏ Hàng Trực Tuyến

Phát triển ứng dụng giỏ hàng trực tuyến cho cửa hàng điện tử, với mảng sản phẩm (mã, tên, giá, số lượng tồn, danh mục), cấu trúc điều kiện để áp dụng mã giảm giá (giảm % hoặc số tiền cố định, có điều kiện về giá trị đơn hàng) và vòng lặp để tính tổng giá trị đơn hàng theo số lượng sản phẩm. Ứng dụng cần tính thuế VAT, phí vận chuyển và hiển thị các phương thức thanh toán.