LẬP TRÌNH PHP

TH.S NGUYỄN ĐÌNH HOÀNG

Mail: hoangnd@itc.edu.vn

NỘI DUNG

- 1. Giới thiệu
- 2. Cấu trúc PHP
- 3. Kiểu dữ liệu, hằng và biến
- 4. Các phép toán trong PHP
- 5. Các cấu trúc điều khiển
- 6. Hàm trong PHP
- 7. Mảng (array)

❑ 5. Các cấu trúc điều khiển

Mệnh đề if trong php được sử dụng để kiểm tra giá trị dạng boolean của điều kiện. Mệnh đề này trả về giá trị **True** hoặc **False** . Có các kiểu của mệnh đề if-else trong php như sau:

- **Mệnh đề if**
- **Mệnh đề if-else**
- **Mệnh đề if-else-if**
- **Mệnh đề if lồng nhau**

❑ 5. Các cấu trúc điều khiển

1. Mệnh đề if

- Mệnh đề if được sử dụng để kiểm tra giá trị dạng boolean của điều kiện. Khối lệnh sau if được thực thi nếu giá trị của điều kiện là **True**
- Cú pháp:

```
if (condition) {
   // khối lệnh này thực thi
   // nếu condition = true
}
```

```
<?php
$age = 20;
if ($age > 18) {
   echo "Tuổi lớn hơn 18";
}
?>
```

❑ 5. Các cấu trúc điều khiển

2. Mệnh đề if-else

- Mệnh đề if-else cũng kiểm tra giá trị dạng boolean của điều kiện. Nếu giá trị điều kiện là **True** thì chỉ có khối lệnh sau if sẽ được thực hiện, nếu là **False** thì chỉ có khối lệnh sau else được thực hiện.
- Cú pháp:

```
if (condition) {
   // khối lệnh này được thực thi
   // nếu condition = true
} else {
   // khối lệnh này được thực thi
   // nếu condition = false
}
```

```
<?php
$number = 13;
if ($number % 2 == 0) {
   echo "Số $number là số chẵn.";
} else {
   echo "Số $number là số lẻ.";
}
?>
```

❑ 5. Các cấu trúc điều khiển

2. Mệnh đề if-else

- Mệnh đề if-else cũng kiểm tra giá trị dạng boolean của điều kiện. Nếu giá trị điều kiện là **True** thì chỉ có khối lệnh sau if sẽ được thực hiện, nếu là **False** thì chỉ có khối lệnh sau else được thực hiện.
- Ví dụ:

```php
<?php
$number = 13;
if ($number % 2 == 0) {
    echo "Số $number là số chẵn.";
} else {
    echo "Số $number là số lẻ.";
}
?>
```

5. Các cấu trúc điều khiển

3. Mệnh đề if-else-if

- Mệnh đề if-else-if cũng kiểm tra giá trị dạng boolean của điều kiện. Nếu giá trị điều kiện if là **True** thì chỉ có khối lệnh sau if sẽ được thực hiện. Nếu giá trị điều kiện if else nào là **True** thì chỉ có khối lệnh sau else if đó sẽ được thực hiện... Nếu tất cả điều kiện của if và else if là **False** thì chỉ có khối lệnh sau else sẽ được thực hiện.
- Cú pháp:

```
if (condition1) {
   // khối lệnh này được thực thi
   // nếu condition1 là true
} else if (condition2) {
   // khối lệnh này được thực thi
   // nếu condition2 là true
}  else if (condition3) {
   // khối lệnh này được thực thi
   // nếu condition3 là true
}
...
else {
   // khối lệnh này được thực thi
   // nếu tất cả những điều kiện trên là false
}
```

5. Các cấu trúc điều khiển

3. Mệnh đề if-else-if

- Mệnh đề if-else-if cũng kiểm tra giá trị dạng boolean của điều kiện. Nếu giá trị điều kiện if là **True** thì chỉ có khối lệnh sau if sẽ được thực hiện. Nếu giá trị điều kiện if else nào là **True** thì chỉ có khối lệnh sau else if đó sẽ được thực hiện... Nếu tất cả điều kiện của if và else if là **False** thì chỉ có khối lệnh sau else sẽ được thực hiện.
- Ví dụ :

```php
<?php
$diem = 75;
if (($diem < 50)) {
    echo “xep loai yeu!";
} else if ($diem >= 80 && $diem < 90) {
    echo "Xep loai B";
} else if ($diem >= 90 && $diem < 100) {
    echo "xep loai A";
} else {
    echo "Xep loai TB";
}
?>
```

5. Các cấu trúc điều khiển

4. Mệnh đề if lồng nhau

Một câu lệnh If hoặc ElseIf bên trong câu lệnh If hoặc ElseIf khác được biết đến như là mệnh đề if lồng nhau. Các câu lệnh if bên trong được thực thi dựa trên các câu lệnh if bên ngoài.

```php
<?php
$number = 23;
if ($number > 0) {
   echo "Number la mot so duong";
   echo "<br>";
   if ($number == 1) {
      echo "Number = 1";
   } else if ($number == 2) {
      echo "Number = 2";
   } else if ($number == 3) {
      echo "Number = 3";
   } else {
      echo "Number khong phai la 0,1,2 hoac 3";
   }
} else if ($number < 0) {
   echo "Number la mot so am";
} else {
   echo "Number la so 0";
}
?>
```

Kết quả:

```
Number la mot so duong
Number khong phai la 0,1,2 hoac 3
```

❑ 5. Các cấu trúc điều khiển

Khái niệm lệnh *"switch case"* trong PHP.

- Lệnh **switch-case** dùng để xác định một danh sách các trường hợp, trong mỗi trường hợp sẽ có một đoạn mã. Khi giá trị của bạn trùng khớp với trường hợp nào thì đoạn mã của trường hợp đó sẽ được thực thi.
- Mệnh đề **switch-case** trong PHP được sử dụng để thực thi 1 hoặc nhiều khối lệnh từ nhiều điều kiện.

5. Các cấu trúc điều khiển

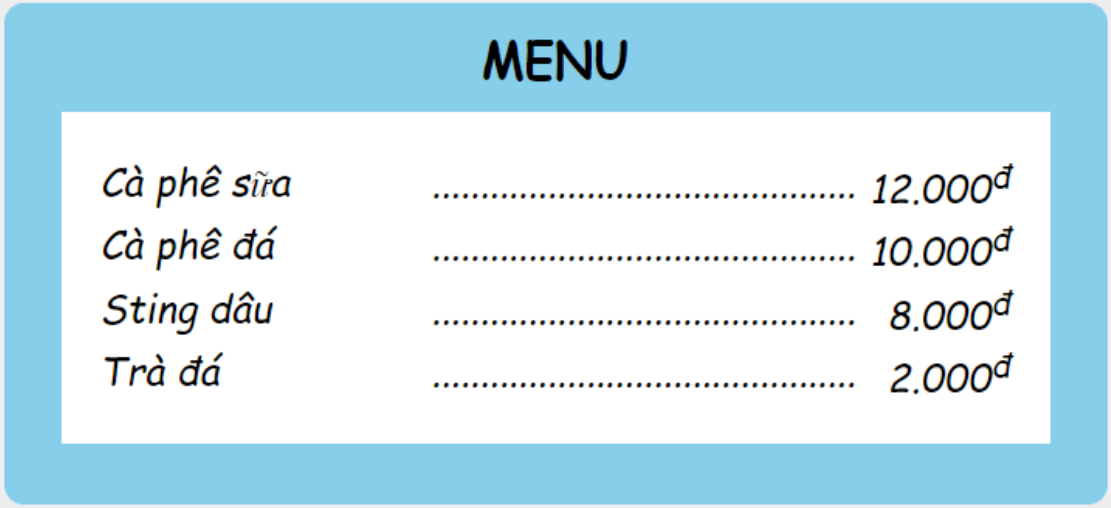
- Cú pháp:

```
switch (bieu_thuc) {
case gia_tri_1:
   // Khối lệnh 1
   break;  //tùy chọn
case gia_tri_2:
   // Khối lệnh 2
   break;  //tùy chọn
......
case gia_tri_n:
   // Khối lệnh n
   break;  //tùy chọn
default:
   // Khối lệnh này được thực thi
   // nếu tất cả các điều kiện trên không thỏa mãn
}
```

5. Các cấu trúc điều khiển

- Để giúp bạn dễ hình dung hơn về khái niệm trên thì tôi có một ví dụ minh họa như sau:

- *Giả thuyết:* Khi đến một quán nước, ở đó có một cái menu giống bên dưới và trong tay bạn chỉ có đúng mười nghìn.



- *Câu hỏi:* Nếu yêu cầu chọn một món nước uống có giá bằng đúng với số tiền mà bạn đang có, thì món nước uống đó là món gì !?

- *Trả lời:* Cà phê đá.

5. Các cấu trúc điều khiển

- Phía trên là một ví dụ mô tả gần giống với lệnh **switch case**, trong đó:
  - Số tiền mà bạn đang có chính là giá trị *(tạm gọi như vậy)*
  - Giá tiền của từng loại nước uống trong menu chính là trường hợp
  - Món nước uống chính là đoạn mã được thực thi.
- Nếu ta chuyển ví dụ trên về dạng mã lệnh trong PHP thì nó sẽ có dạng như sau:

```php
<?php
	$money = 10000;
	switch ($money)
	{
		case 2000:
		echo "Trà đá"; break;
		case 8000:
		echo "Sting dâu"; break;
		case 10000:
		echo "Cà phê đá"; break;
		case 12000:
		echo "Cà phê sữa"; break;
	}
?>
```

❑ 5. Các cấu trúc điều khiển

Mệnh đề Switch-case khi không sử dụng 'break'

- Khi không sử dụng từ khóa 'break' trong mệnh đề switch-case. Điều này có nghĩa là các khối lệnh sau case có giá trị phù hợp sẽ được thực thi.
- Ví dụ về mệnh đề switch-case:

```php
<?php
    $num=20;
    switch($num){
      case 10:
        echo("number is equals to 10");
      case 20:
        echo("number is equal to 20");
      case 30:
        echo("number is equal to 30");
      default:
        echo("Not in 10, 20 or 30");
    }
?>
```

❑ 5. Các cấu trúc điều khiển

Tầm quan trọng của lệnh *"break"*

- Trong danh sách các trường hợp của lệnh **switch case**, khi giá trị của bạn trùng khớp với trường hợp nào thì đoạn mã của trường hợp đó sẽ được thực, ngoài ra các đoạn mã của những trường hợp nằm bên dưới trường hợp trùng khớp cũng sẽ được thực thi luôn.
- Từ đây, lệnh **break** giúp ta ngăn chặn việc thực thi các đoạn mã của những trường hợp nằm bên dưới trường hợp trùng khớp.

5. Các cấu trúc điều khiển

Ví dụ, lệnh switch case không sử dụng break

```
<?php

   $season = "ha";

   switch ($season){
      case "xuan":
         echo "<p>Mùa Xuân</p>";
      case "ha":
         echo "<p>Mùa Hạ</p>";
      case "thu":
         echo "<p>Mùa Thu</p>";
      case "dong":
         echo "<p>Mùa Đông</p>";
   }

?>
```

Ví dụ, lệnh switch case có sử dụng break

```
<?php

   $season = "ha";

   switch ($season){
      case "xuan":
         echo "<p>Mùa Xuân</p>";
         break;
      case "ha":
         echo "<p>Mùa Hạ</p>";
         break;
      case "thu":
         echo "<p>Mùa Thu</p>";
         break;
      case "dong":
         echo "<p>Mùa Đông</p>";
         break;
   }

?>
```

5. Các cấu trúc điều khiển

Công dụng của lệnh *"default"*

- Lệnh **default** dùng để xác định một đoạn mã mặc định sẽ được thực thi khi giá trị của bạn không trùng khớp với bất kỳ trường hợp nào.

```php
<?php
  $season = "abc";
  switch ($season){
    case "xuan":
        echo "Mùa Xuân";
        break;
    case "ha":
        echo "Mùa Hạ";
        break;
    case "thu":
        echo "Mùa Thu";
        break;
    case "dong":
        echo "Mùa Đông";
        break;
    default:
        echo "KHÔNG XÁC ĐỊNH";
        break;
  }
?>
```

5. Các cấu trúc điều khiển

- Mệnh đề **switch-case** trong PHP được sử dụng để thực thi 1 hoặc nhiều khối lệnh từ nhiều điều kiện.
- Ví dụ :

```php
<?php
$num=20;
switch($num){
   case 10:
      echo("number is equals to 10");
      break;
   case 20:
      echo("number is equal to 20");
      break;
   case 30:
      echo("number is equal to 30");
      break;
   default:
      echo("Not in 10, 20 or 30");

}
?>
```

5. Các cấu trúc điều khiển

- Ví dụ : Hôm nay là thứ mấy.

```php
<?php
  $date = getdate();
  switch ($date["weekday"]){
      case "Monday":
          echo "Thứ hai";
          break;
      case "Tuesday":
          echo "Thứ ba";
          break;
      case "Wednesday":
          echo "Thứ tư";
          break;
      case "Thursday":
          echo "Thứ năm";
          break;
      case "Friday":
          echo "Thứ sáu";
          break;
      case "Saturday":
          echo "Thứ bảy";
          break;
      case "Sunday":
          echo "Chủ Nhật";
          break;
  }
?>
```

❑ 5. Các cấu trúc điều khiển

Nhóm các trường hợp lại với nhau

- Nếu trong danh sách các trường hợp của lệnh **switch case** có những trường hợp mà bạn muốn cùng thực thi một đoạn mã thì ta hãy nhóm các trường hợp đó lại với nhau.

```php
<?php
  $month = 5;
  switch ($month){
      case 1:
      case 2:
      case 3:
          echo "Mùa Xuân";
          break;
      case 4:
      case 5:
      case 6:
          echo "Mùa Hạ";
          break;
      case 7:
      case 8:
      case 9:
          echo "Mùa Thu";
          break;
      case 10:
      case 11:
      case 12:
          echo "Mùa Đông";
          break;
  }
?>
```

❑ 5. Các cấu trúc điều khiển

Lệnh *"switch case lồng nhau"*

- Thật ra, lệnh switch case lồng nhau chỉ là cách sử dụng nâng cao của lệnh switch case thông thường, nó giúp ta mở rộng phạm vi xét duyệt các trường hợp. Từ đó, chọn được đoạn mã thích hợp nhất để thực thi.

```php
<?php
  $season = “Xuân";
  $month = 3;
  switch($season){
     case "Xuân":
          switch($month){
               case 1:
                    echo "Mùa Xuân - Tháng 1";
                    break;
               case 2:
                    echo "Mùa Xuân - Tháng 2";
                    break;
               case 3:
                    echo "Mùa Xuân - Tháng 3";
                    break;
               default:
                    echo "Mùa " + $season + "
                      không có tháng" + $month;
                    break;
          }
          break;
```

❑ 5. Các cấu trúc điều khiển

Lệnh *"switch case lồng nhau"*

- Thật ra, lệnh switch case lồng nhau chỉ là cách sử dụng nâng cao của lệnh switch case thông thường, nó giúp ta mở rộng phạm vi xét duyệt các trường hợp. Từ đó, chọn được đoạn mã thích hợp nhất để thực thi.

```
case "Hạ":
        switch($month){
            case 4:
                echo "Mùa Hạ - Tháng 4";
                break;
            case 5:
                echo "Mùa Hạ - Tháng 5";
                break;
            case 6:
                echo "Mùa Hạ - Tháng 6";
                break;
            default:
                echo "Mùa " + $season + “
                  không có tháng" + $month;
                break;
        }
        break;
```

5. Các cấu trúc điều khiển

```
case "Thu":
        switch($month){
            case 7:
                echo "Mùa Thu - Tháng 7";
                break;
            case 8:
                echo "Mùa Thu - Tháng 8";
                break;
            case 9:
                echo "Mùa Thu - Tháng 9";
                break;
            default:
                echo "Mùa " + $season + “
                  không có tháng" + $month;
                break;
        }
        break;
```

```
case "Đông":
        switch($month){
            case 10:
                echo "Mùa Đông - Tháng 10";
                break;
            case 11:
                echo "Mùa Đông - Tháng 11";
                break;
            case 12:
                echo "Mùa Đông - Tháng 12";
                break;
            default:
                echo "Mùa " + $season + “
                  không có tháng" + $month;
                break;
        }
        break;
    default:
        echo "KHÔNG XÁC ĐỊNH";
        break;
  }
?>
```

5. Các cấu trúc điều khiển

Vòng lặp for đơn giản

- Vòng lặp for đơn giản giống như trong C/C++/Java. Chúng ta có thể khởi tạo biến, kiểm tra điều kiện và tăng/giảm giá trị của biến.
- Cú pháp:

```
for(biểu thức 1; biểu thức 2; biểu thức 3)
{
        //Đoạn mã mà bạn muốn được thực thi
}
```

Trong đó:

- Biểu thức 1 thường là một câu lệnh khai báo biến

(biến này dùng để tham gia vào biểu thức 2)

- Biểu thức 2 là một biểu thức điều kiện

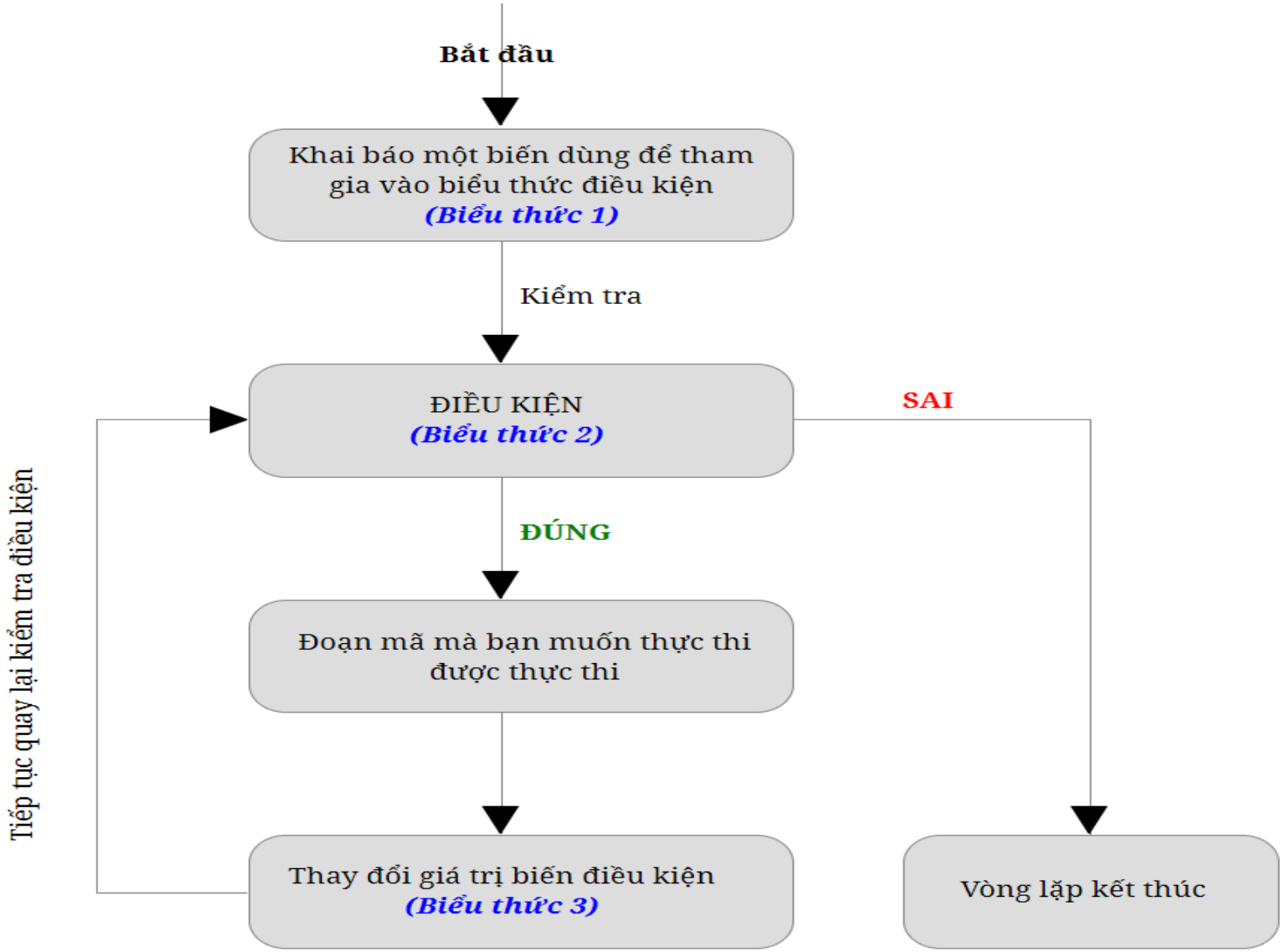
(nếu điều kiện đúng thì đoạn mã sẽ được thực thi, còn nếu điều kiện sai thì vòng lặp kết thúc)

- Biểu thức 3 thường là một biểu thức làm thay đổi giá trị của biến được khai báo trong biểu thức 1

(mục đích là để cho biểu thức điều kiện dần trở nên bị SAI, giúp vòng lặp được kết thúc)

5. Các cấu trúc điều khiển

Dưới đây là sơ đồ minh họa cho nguyên lý hoạt động của vòng lặp for:



5. Các cấu trúc điều khiển

Ví dụ: Sử dụng vòng lặp for để hiển thị một dãy số tăng dần từ 1 đến 10.

```
<?php
    for($i=0;$n<10;$i++){
       echo  " Số: ". ($i+1)."<br/>";
    }
?>
```

Dưới đây là phần mô tả các bước thực thi của vòng lặp trên:

•Lần thứ nhất

- Khai báo biến i với giá trị là 0.
- Kiểm tra xem điều kiện i < 10 có đúng hay không.

(Kết quả đúng nên đoạn mã nằm trong cặp dấu {} được thực thi)

- Giá trị của biến i được tăng thêm một *(tức bây giờ giá trị của biến i sẽ là 1)*

•Lần thứ hai

- Kiểm tra xem điều kiện i < 10 có đúng hay không.

(Kết quả đúng nên đoạn mã nằm trong cặp dấu {} được thực thi)

- Giá trị của biến i được tăng thêm một *(tức bây giờ giá trị của biến i sẽ là 2)*

5. Các cấu trúc điều khiển

Ví dụ: Sử dụng vòng lặp for để hiển thị một dãy số tăng dần từ 1 đến 10.

```
<?php
    for($i=0;$n<10;$i++){
       echo  " Số: ". ($i+1)."<br/>";
    }
?>
```

•....

•Lần thứ mười

- Kiểm tra xem điều kiện i < 10 có đúng hay không.

(Kết quả đúng nên đoạn mã nằm trong cặp dấu {} được thực thi)

- Giá trị của biến i được tăng thêm một *(tức bây giờ giá trị của biến i sẽ là 10)*

•Lần thứ mười một

- Kiểm tra xem điều kiện i < 10 có đúng hay không.

  *(Kết quả là **SAI** vì 10 không nhỏ hơn 10)*

  ==> VÒNG LẶP KẾT THÚC

5. Các cấu trúc điều khiển

Ví dụ: - Sử dụng vòng lặp for để hiển thị một dãy số giảm dần từ 9 xuống 2.

```
<?php
    for($i=9;$n>1;$i--){
       echo  " Số: ". $i."<br/>";
    }
?>
```

Dưới đây là phần mô tả các bước thực thi của vòng lặp trên:

•Lần thứ nhất

- Khai báo biến i với giá trị là 9.
- Kiểm tra xem điều kiện i > 1 có đúng hay không.

(Kết quả đúng nên đoạn mã nằm trong cặp dấu {} được thực thi)

- Giá trị của biến i bị giảm đi một *(tức bây giờ giá trị của biến i sẽ là 8)*

•Lần thứ hai

- Kiểm tra xem điều kiện i > 1 có đúng hay không.

(Kết quả đúng nên đoạn mã nằm trong cặp dấu {} được thực thi)

- Giá trị của biến i bị giảm đi một *(tức bây giờ giá trị của biến i sẽ là 7)*

5. Các cấu trúc điều khiển

Ví dụ: - Sử dụng vòng lặp for để hiển thị một dãy số giảm dần từ 9 xuống 2.

```
<?php
    for($i=9;$n>1;$i--){
       echo  " Số: ". $i."<br/>";
    }
?>
```

- ....
- Lần thứ tám
  - Kiểm tra xem điều kiện i > 1 có đúng hay không.

(Kết quả đúng nên đoạn mã nằm trong cặp dấu {} được thực thi)

  - Giá trị của biến i bị giảm đi một *(tức bây giờ giá trị của biến i sẽ là 1)*

- Lần thứ chín
  - Kiểm tra xem điều kiện i > 1 có đúng hay không.

*(Kết quả là **SAI** vì 1 không lớn hơn 1)*

==> VÒNG LẶP KẾT THÚC

❑ 5. Các cấu trúc điều khiển

Vòng lặp for lồng nhau

- Thực chất vòng lặp for lồng nhau chỉ là cách sử dụng nâng cao của vòng lặp for thông thường để giúp cho số lần lặp được tăng theo cấp số nhân.
- Vòng lặp con được đặt vào bên trong vòng lặp cha. Khi điều kiện của vòng lặp cha đúng thì vòng lặp con sẽ được thực thi.
- Dưới đây là cú pháp cơ bản của một vòng lặp for lồng nhau:

```
for(biểu thức 1; biểu thức 2; biểu thức 3){
        for(biểu thức 1; biểu thức 2; biểu thức 3){
        //Đoạn mã mà bạn muốn được thực thi
         }
}
```

5. Các cấu trúc điều khiển

Vòng lặp for lồng nhau

- Chúng ta có thể sử dụng vòng lặp for bên trong cho vòng lặp for trong PHP, nó được gọi là vòng lặp for lồng nhau .
- Ví dụ:

```
<?php
for($i=1;$i<=3;$i++){
   for($j=1;$j<=3;$j++){
      echo "$i   $j<br/>";
   }
}
?>
```

Kết quả:

```
1  1
1  2
1  3
2  1
2  2
2  3
3  1
3  2
3  3
```

❑ 5. Các cấu trúc điều khiển

For-each trong PHP

- Vòng lặp foreach chỉ làm việc với mảng (Array)
- Vòng lặp foreach dùng để lặp lại việc thực thi một đoạn mã nào đó với số lần lặp lại bằng với số phần tử của mảng (Ví dụ: nếu mảng có 5 phần tử thì đoạn mã sẽ được thực thi lặp lại 5 lần)
- Trong mỗi lần lặp, giá trị của phần tử mảng hiện tại sẽ được lưu vào một biến.
- Cú pháp:

```
foreach ($array as $var) {
   // Khối lệnh được thực thi
}
```

❑ 5. Các cấu trúc điều khiển

For-each trong PHP

- Ví dụ:

```php
<?php
$season=array("summer","winter","spring","autumn");
foreach( $season as $arr ){
   echo "Season is: $arr<br>";
}
?>
```

Kết quả:

```
Season is: summer
Season is: winter
Season is: spring
Season is: autumn
```

5. Các cấu trúc điều khiển

Vòng lặp while trong php

- **Vòng lặp while trong php** được sử dụng để lặp một phần của chương trình một vài lần. Nếu số lần lặp không được xác định trước thì vòng lặp lặp while được khuyến khích sử dụng trong trường hợp này.
- **Cú pháp:**

```
while(condition) {
   // Khối lệnh được lặp lại cho đến khi condition = False
}
```

```
<?php
$i = 1;
while ($i <= 10) {
   echo "$i <br>";
   $i++;
}
?>
```

❑ 5. Các cấu trúc điều khiển

Vòng lặp while vô tận

- Nếu bạn để điều kiện lặp là True thì vòng lặp while sẽ chạy đến vô tận... Đến khi bạn stop chương trình bằng cách tắt trình duyệt.
- Ví dụ về vòng lặp while vô tận trong php:

```
<?php
while (true) {
    echo "Vòng lặp while vô tận...";
}
?>
```

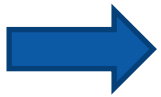
❑ 5. Các cấu trúc điều khiển

Vòng lặp do-while trong php

- **Vòng lặp do-while trong php** được sử dụng để lặp một phần của chương trình một vài lần. Tương tự như vòng lặp while, ngoại trừ do-while thực hiện lệnh ít nhất một lần ngay cả khi điều kiện là False.
- **Cú pháp:**

```
do {
   // Khối lệnh được thực thi
} while(condition);
```

```
<?php
$a = 1;
$sum = 0;
do {
   $sum += $a;
   $a++;
} while ($a <= 5);
echo("Sum of 1 to 5  is " . $sum);
?>
```



Kết quả:

```
Sum of 1 to 5 is 15
```

❑ 5. Các cấu trúc điều khiển

Vòng lặp do-while vô tận

- Nếu bạn để điều kiện lặp là True thì vòng lặp do-while sẽ chạy đến vô tận... Đến khi bạn stop chương trình bằng cách tắt trình duyệt.
- Ví dụ về vòng lặp do-while vô tận:

```php
<?php
do {
    echo("Vòng lặp do-while vô tận...");
} while (true);
?>
```

❑ 5. Các cấu trúc điều khiển

Sử dụng Break trong php

- **Từ khóa break trong php** được sử dụng để thoát vòng lặp hoặc trong mệnh đề switch tại điều kiện đã được chỉ định. Đối với vòng lặp bên trong vòng lặp khác, thì nó chỉ thoát vòng lặp bên trong đó.

```
<?php
for ($i = 1; $i <= 10; $i++) {
   if ($i == 5) {
      break;
   }
   echo "$i <br>";
}
?>
```

5. Các cấu trúc điều khiển

Sử dụng Break trong php

- Ví dụ sử dụng break với vòng lặp bên trong vòng lặp for khác:

```
<?php
for ($i = 1; $i <= 3; $i++) {
   for ($j = 1; $j <= 3; $j++) {
      if ($i == 2 && $j == 2) {
         break;
      }
      echo ($i . " " . $j . "<br>");
   }
}
?>
```

❑ 5. Các cấu trúc điều khiển

Sử dụng Continue trong php

- Ví dụ sử dụng Continue với vòng lặp bên trong vòng lặp for khác:

```
<?php
for ($i = 1; $i <= 3; $i++) {
   for ($j = 1; $j <= 3; $j++) {
      if ($i == 2 && $j == 2) {
         continue;
      }
      // Không in trường hợp i=2 và j=2 ra màn hình
      echo ("$i $j <br>");
   }
}
?>
```

❑ 5. Các cấu trúc điều khiển

Sử dụng Continue trong php

- **Từ khóa continue trong php** được sử dụng để tiếp tục vòng lặp tại điều kiện đã được xác định, với điều kiện đó khối lệnh phía sau từ khóa continue sẽ không được thực thi. Đối với vòng lặp bên trong một vòng lặp khác, continue chỉ có tác dụng với vọng lặp bên trong đó.

```
<?php
for ($i = 1; $i <= 10; $i++) {
   if ($i == 5) {
      continue;
   }
   // Khi i == 5 thì không in i = 5 ra màn hình
   echo "$i <br>";
}
?>
```

6. Hàm trong PHP

- Các hàm trong PHP thực sự góp phần tạo nên sự mạnh mẽ của ngôn ngữ lập trình PHP, nó có hơn 1000 hàm tích hợp sẵn.
- Bên cạnh các hàm PHP có sẵn, chúng ta có thể tạo các hàm riêng của chúng ta.
- Một hàm là một khối các câu lệnh có thể được sử dụng nhiều lần trong một chương trình.
- Một hàm sẽ không thực thi ngay lập tức khi một trang tải lên.
- Một hàm sẽ được thực thi bằng một cuộc gọi đến hàm.

6. Hàm trong PHP

Tạo một hàm do người dùng định nghĩa trong PHP

- Khai báo hàm do người dùng định nghĩa bắt đầu bằng từ **function**:
- Cú pháp:

```
function functionName([parameter1]...[,parameterN]) {
        // code được thực thi
        statement[s] ;
        [return ….. ;]
}
```

Trong đó:

- functionName: tên hàm
- parameter: danh sách tham số
- return: giá trị hàm trả về nếu có

Lưu ý: Tên hàm có thể bắt đầu bằng một chữ cái hoặc dấu gạch dưới (không phải là số).

6. Hàm trong PHP

Các đối số của hàm trong PHP

- Dữ liệu có thể được chuyển đến các hàm thông qua các tham số. Một tham số giống như một biến.
- Các tham số được xác định sau tên hàm, bên trong dấu ngoặc đơn. Bạn có thể thêm bao nhiêu tham số tùy thích, chỉ cần tách chúng bằng dấu phẩy.
- Ví dụ sau có một hàm với một đối số ($n)

```php
<?php
    function calFactorial($n)
    {
        $result = 1;
        for($i=2 ; $i<=$n ; $i++)
            $result *=$i;
        return $result;
    }
    $n = 4;
    echo $n.'!= '.calFactorial($n);
?>
```

6. Hàm trong PHP

Các đối số của hàm trong PHP

- Ví dụ sau truyền 2 đối số vào hàm và tính tổng của chúng:

```php
<?php
function sum($a, $b) {
   $tong = $a + $b;
   echo "$a + $b = $tong <br>";
}

sum(1, 2);
sum(1, 3);
sum(10, 20);
sum(10, 30);
?>
```

Kết quả:

```
1 + 2 = 3
1 + 3 = 4
10 + 20 = 30
10 + 30 = 40
```

6. Hàm trong PHP

Giá trị đối số mặc định trong PHP

- Ví dụ sau đây cho thấy cách sử dụng đối số mặc định. Nếu chúng ta gọi hàm setHeight() không có đối số thì giá trị mặc định là đối số:

```php
<?php
function setHeight($minheight = 50) {
   echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // sử dụng giá trị mặc định 50
setHeight(135);
setHeight(80);
?>
```

Kết quả:

```
The height is : 350
The height is : 50
The height is : 135
The height is : 80
```

6. Hàm trong PHP

Giá trị trả về của hàm trong PHP

- Sử dụng câu lệnh **return** để trả về giá trị cho một hàm:

```php
<?php
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

Kết quả:

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

❑ 6. Hàm trong PHP

Hàm tham trị

- Trong PHP cho phép chúng ta truyền biến theo tham trị hay tham chiếu. Truyền theo tham trị là giá trị biến không thay đổi sau khi hàm chạy xong. Trong ví dụ này ta có hàm Hello, ta truyền biến $str có giá trị "Xin chào" bên trong hàm có chuỗi "Tèo". Nhưng nó chỉ Print câu **xin chào**.

```
<?php function hello($str2){
        $str2 .= 'Tèo';
}
$str = 'Xin Chào';
hello($str);
echo $str; ?>
```

6. Hàm trong PHP

Hàm tham chiếu

- Ngược lại truyền theo tham trị, tham số bên trong hàm thay đổi sau khi gọi hàm.

  Cần thêm & trước tham số nào muốn làm tham chiếu. Kết quả hiển thị "**Xin Chào Tèo**"

```php
<?php function hello(&$str2){
        $str2 .= 'Tèo';
}
$str = 'Xin Chào';
hello($str);
echo $str; ?>
```

6. Hàm trong PHP

Tham số có độ dài biến đổi trong PHP

- PHP hỗ trợ chức năng tham số (đối số) có độ dài biến đổi. Nó có nghĩa là bạn có thể chuyền 0, 1 hoặc n đối số trong hàm. Để làm như vậy, bạn cần sử dụng 3 dấu ba chấm (...) trước tên đối số.
- Khái niệm 3 dấu chấm được thực hiện cho đối số có chiều dài thay đổi kể từ PHP 5.6.

```
<?php
function add(...$numbers) {
   $sum = 0;
   foreach ($numbers as $n) {
      $sum += $n;
   }
   return $sum;
}
echo add(1, 2, 3, 4);
?>
```

6. Hàm trong PHP

Hàm đệ quy trong PHP

- Một hàm được gọi là đệ quy nếu bên trong thân nó có một lời gọi đến chính nó. Nghe có vẻ vô lý nhỉ ? Một hàm làm sao có thể gọi nó mãi được, vì nếu như vậy sẽ sinh ra một vòng lặp vô tận. Nhưng trong thực tế, một hàm đệ quy luôn có điều kiện đừng được gọi là "điểm neo". Khi đạt tới điểm neo, hàm sẽ không gọi chính nó nữa.
- Cú pháp:

```
return_type function_name() {
    // code
    function_name();
}
```

6. Hàm trong PHP

- Ví dụ về đệ quy **vòng lặp vô tận** trong php
- Bạn nên ngừng load trang hoặc tắt trình duyệt tránh bị đơ máy vì vòng lặp vô hạn.

```
<?php
function p() {
    echo "hello <br>";
    p();
}
p();
?>
```

6. Hàm trong PHP

Ví dụ: Đệ quy vòng lặp có điểm dừng "**điểm neo**"

```php
<?php
function giaithua($n) {
   if ($n == 1)
   return 1;
   else
   return ($n * giaithua($n - 1));
}
// tính giai thừa của 5
echo "Giai thừa của 5 là:" . giaithua(5);
?>
```

Chương trình trên hoạt động như sau:

```
1   giaithua(5)
2      giaithua(4)
3         giaithua(3)
4            giaithua(2)
5               giaithua(1)
6                  return 1
7               return 2*1 = 2
8            return 3*2 = 6
9         return 4*6 = 24
10     return 5*24 = 120
```

KẾT THÚC