[integer](https://www.php.net/manual/zh/language.types.integer.php) 是集合 ℤ = {..., -2, -1, 0, 1, 2, ...} 中的某个数。

参见：

* [任意长度整数／GMP](https://www.php.net/manual/zh/book.gmp.php)
* [浮点型](https://www.php.net/manual/zh/language.types.float.php)
* [任意精度数学库／BCMath](https://www.php.net/manual/zh/book.bc.php)

**语法**

整型值可以使用十进制，十六进制，八进制或二进制表示，前面可以加上可选的符号（- 或者 +）。

二进制表达的 [integer](https://www.php.net/manual/zh/language.types.integer.php) 自 PHP 5.4.0 起可用。

要使用八进制表达，数字前必须加上 *0*（零）。要使用十六进制表达，数字前必须加上 *0x*。要使用二进制表达，数字前必须加上 *0b*。

**Example #1 整数文字表达**

<?php  
$a = 1234; // 十进制数  
$a = -123; // 负数  
$a = 0123; // 八进制数 (等于十进制 83)  
$a = 0x1A; // 十六进制数 (等于十进制 26)  
$a = 0b11111111; // 二进制数字 (等于十进制 255)  
?>

[integer](https://www.php.net/manual/zh/language.types.integer.php) 语法的结构形式是：

decimal : [1-9][0-9]\*

| 0

hexadecimal : 0[xX][0-9a-fA-F]+

octal : 0[0-7]+

binary : 0b[01]+

integer : [+-]?decimal

| [+-]?hexadecimal

| [+-]?octal

| [+-]?binary

整型数的字长和平台有关，尽管通常最大值是大约二十亿（32 位有符号）。64 位平台下的最大值通常是大约 9E18，除了 Windows 下 PHP 7 以前的版本，总是 32 位的。 PHP 不支持无符号的 [integer](https://www.php.net/manual/zh/language.types.integer.php)。[Integer](https://www.php.net/manual/zh/language.types.integer.php) 值的字长可以用常量 **PHP\_INT\_SIZE**来表示，自 PHP 4.4.0 和 PHP 5.0.5后，最大值可以用常量 **PHP\_INT\_MAX** 来表示，最小值可以在 PHP 7.0.0 及以后的版本中用常量 **PHP\_INT\_MIN** 表示。

**Warning**

PHP 7 以前的版本里，如果向八进制数传递了一个非法数字（即 8 或 9），则后面其余数字会被忽略。PHP 7 以后，会产生 Parse Error。

**整数溢出**

如果给定的一个数超出了 [integer](https://www.php.net/manual/zh/language.types.integer.php) 的范围，将会被解释为 [float](https://www.php.net/manual/zh/language.types.float.php)。同样如果执行的运算结果超出了 [integer](https://www.php.net/manual/zh/language.types.integer.php) 范围，也会返回 [float](https://www.php.net/manual/zh/language.types.float.php)。

**Example #2 32 位系统下的整数溢出**

<?php  
$large\_number = 2147483647;  
var\_dump($large\_number);                     // int(2147483647)  
  
$large\_number = 2147483648;  
var\_dump($large\_number);                     // float(2147483648)  
  
$million = 1000000;  
$large\_number =  50000 \* $million;  
var\_dump($large\_number);                     // float(50000000000)  
?>

**Example #3 64 位系统下的整数溢出**

<?php  
$large\_number = 9223372036854775807;  
var\_dump($large\_number);                     // int(9223372036854775807)  
  
$large\_number = 9223372036854775808;  
var\_dump($large\_number);                     // float(9.2233720368548E+18)  
  
$million = 1000000;  
$large\_number =  50000000000000 \* $million;  
var\_dump($large\_number);                     // float(5.0E+19)  
?>

PHP 中没有整除的运算符。*1/2* 产生出 [float](https://www.php.net/manual/zh/language.types.float.php) *0.5*。 值可以舍弃小数部分，强制转换为 [integer](https://www.php.net/manual/zh/language.types.integer.php)，或者使用 [round()](https://www.php.net/manual/zh/function.round.php) 函数可以更好地进行四舍五入。

<?php  
var\_dump(25/7);         // float(3.5714285714286)   
var\_dump((int) (25/7)); // int(3)  
var\_dump(round(25/7));  // float(4)   
?>

**转换为整型**

要明确地将一个值转换为 [integer](https://www.php.net/manual/zh/language.types.integer.php)，用 *(int)* 或 *(integer)* 强制转换。不过大多数情况下都不需要强制转换，因为当运算符，函数或流程控制需要一个 [integer](https://www.php.net/manual/zh/language.types.integer.php) 参数时，值会自动转换。还可以通过函数 [intval()](https://www.php.net/manual/zh/function.intval.php) 来将一个值转换成整型。

将 [resource](https://www.php.net/manual/zh/language.types.resource.php) 转换成 [integer](https://www.php.net/manual/zh/language.types.integer.php) 时， 结果会是 PHP 运行时为 [resource](https://www.php.net/manual/zh/language.types.resource.php) 分配的唯一资源号。

参见：[类型转换的判别](https://www.php.net/manual/zh/language.types.type-juggling.php)。

**从[布尔值](https://www.php.net/manual/zh/language.types.boolean.php)转换**

**FALSE** 将产生出 *0*（零），**TRUE** 将产生出 *1*（壹）。

**从**[**浮点型**](https://www.php.net/manual/zh/language.types.float.php)**转换**

当从浮点数转换成整数时，将*向下*取整。

如果浮点数超出了整数范围（32 位平台下通常为 *+/- 2.15e+9 = 2^31*，64 位平台下，除了 Windows，通常为 *+/- 9.22e+18 = 2^63*），则结果为未定义，因为没有足够的精度给出一个确切的整数结果。在此情况下没有警告，甚至没有任何通知！

**Note**:

PHP 7.0.0 起，NaN 和 Infinity 在转换成 [integer](https://www.php.net/manual/zh/language.types.integer.php) 时，不再是 undefined 或者依赖于平台，而是都会变成零。

**Warning**

绝不要将未知的分数强制转换为 [integer](https://www.php.net/manual/zh/language.types.integer.php)，这样有时会导致不可预料的结果。

<?php  
echo (int) ( (0.1+0.7) \* 10 ); // 显示 7!  
?>

参见[关于浮点数精度的警告](https://www.php.net/manual/zh/language.types.float.php#warn.float-precision)。

**从字符串转换**

参见[字符串转换为数值](https://www.php.net/manual/zh/language.types.string.php#language.types.string.conversion)。

**从其它类型转换**

**Caution**

没有定义从其它类型转换为整型的行为。*不要*依赖任何现有的行为，因为它会未加通知地改变。