BCMath Arbitrary Precision Mathematics

Introduction

For arbitrary precision mathematics PHP offers the Binary Calculator which supports numbers of any size and precision, represented as strings.

Installing/Configuring

Requirements

Since PHP 4.0.4, libbcmath is bundled with PHP. You don't need any external libraries for this extension.

Installation

These functions are only available if PHP was configured with *--enable-bcmath*.

The Windows version of PHP has built-in support for this extension. You do not need to load any additional extensions in order to use these functions.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

BC math configuration options

Name	Default	Changeable	Changelog
bcmath.scale	"0"	PHP_INI_ALL	

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

`bcmath.scale` integer
: Number of decimal digits for all bcmath functions. See also bcscale().

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

BC Math Functions

bcadd

bcadd -- Add two arbitrary precision numbers

Description

string **bcadd** (string `$left_operand`, string `$right_operand` [, int `$scale`])

Sums *`left_operand`* and *`right_operand`*.

Parameters

`left_operand`
: The left operand, as a string.

`right_operand`
: The right operand, as a string.

`scale`
: This optional parameter is used to set the number of digits after the decimal place in the result. You can also set the global default scale for all functions by using bcscale().

Return Values

The sum of the two operands, as a string.

Examples

Example #1 - bcadd() example

```
<?php

$a = '1.234';
$b = '5';

echo bcadd($a, $b);     // 6
echo bcadd($a, $b, 4);  // 6.2340

?>
```

See Also

- bcsub()

bccomp

bccomp -- Compare two arbitrary precision numbers

Description

int **bccomp** (string `$left_operand`, string `$right_operand` [, int `$scale`])

Compares the *`left_operand`* to the *`right_operand`* and returns the result as an integer.

Parameters

`left_operand`
: The left operand, as a string.

`right_operand`
: The right operand, as a string.

`scale`
: The optional *`scale`* parameter is used to set the number of digits after the decimal place which will be used in the comparison.

Return Values

Returns 0 if the two operands are equal, 1 if the *`left_operand`* is larger than the *`right_operand`*, -1 otherwise.

Examples

Example #2 - bccomp() example

```
<?php

echo bccomp('1', '2') . "\n";   // -1
echo bccomp('1.00001', '1', 3); // 0
echo bccomp('1.00001', '1', 5); // 1

?>
```

bcdiv

bcdiv -- Divide two arbitrary precision numbers

Description

string **bcdiv** (string `$left_operand`, string `$right_operand` [, int `$scale`])

Divides the *`left_operand`* by the *`right_operand`*.

Parameters

`left_operand`
: The left operand, as a string.

`right_operand`
: The right operand, as a string.

`scale`
: This optional parameter is used to set the number of digits after the decimal place in the result. You can also set the global default scale for all functions by using bcscale().

Return Values

Returns the result of the division as a string, or **NULL** if *`right_operand`* is 0.

Examples

Example #3 - bcdiv() example

```
<?php

echo bcdiv('105', '6.55957', 3);  // 16.007

?>
```

See Also

- bcmul()

bcmod

bcmod -- Get modulus of an arbitrary precision number

Description

string **bcmod** (string `$left_operand`, string `$modulus`)

Get the modulus of the *`left_operand`* using *`modulus`*.

Parameters

`left_operand`
: The left operand, as a string.

`modulus`
: The modulus, as a string.

Return Values

Returns the modulus as a string, or **NULL** if *`modulus`* is 0.

Examples

Example #4 - bcmod() example

```
<?php
echo bcmod('4', '2'); // 0
echo bcmod('2', '4'); // 2
?>
```

See Also

- bcdiv()

bcmul

bcmul -- Multiply two arbitrary precision number

Description

string **bcmul** (string `$left_operand`, string `$right_operand` [, int `$scale`])

Multiply the *`left_operand`* by the *`right_operand`*.

Parameters

`left_operand`
: The left operand, as a string.

`right_operand`
: The right operand, as a string.

`scale`
: This optional parameter is used to set the number of digits after the decimal place in the result. You can also set the global default scale for all functions by using bcscale().

Return Values

Returns the result as a string.

Examples

Example #5 - bcmul() example

```
<?php
echo bcmul('1.34747474747', '35', 3); // 47.161
echo bcmul('2', '4'); // 8
?>
```

See Also

- bcdiv()

bcpow

bcpow -- Raise an arbitrary precision number to another

Description

string **bcpow** (string `$left_operand`, string `$right_operand` [, int `$scale`])

Raise *`left_operand`* to the power *`right_operand`*.

Parameters

`left_operand`
: The left operand, as a string.

`right_operand`
: The right operand, as a string.

`scale`
: This optional parameter is used to set the number of digits after the decimal place in the result. You can also set the global default scale for all functions by using bcscale().

Return Values

Returns the result as a string.

Examples

Example #6 - bcpow() example

```
<?php

echo bcpow('4.2', '3', 2); // 74.08

?>
```

See Also

- bcpowmod()
- bcsqrt()

bcpowmod

bcpowmod -- Raise an arbitrary precision number to another, reduced by a specified modulus

Description

string **bcpowmod** (string `$left_operand`, string `$right_operand`, string `$modulus` [, int `$scale`])

Use the fast-exponentiation method to raise *`left_operand`* to the power *`right_operand`* with respect to the modulus *`modulus`*.

Parameters

`left_operand`
: The left operand, as a string.

`right_operand`
: The right operand, as a string.

`modulus`
: The modulus, as a string.

`scale`
: This optional parameter is used to set the number of digits after the decimal place in the result. You can also set the global default scale for all functions by using bcscale().

Return Values

Returns the result as a string, or **NULL** if *`modulus`* is 0.

Notes

Note
Because this method uses the modulus operation, non-natural numbers may give unexpected results. A natural number is any positive non-zero integer.

Examples

The following two statements are functionally identical. The bcpowmod() version however, executes in less time and can accept larger parameters.

```php
<?php
$a = bcpowmod($x, $y, $mod);

$b = bcmod(bcpow($x, $y), $mod);

// $a and $b are equal to each other.

?>
```

See Also

- bcpow()
- bcmod()

bcscale

bcscale -- Set default scale parameter for all bc math functions

Description

bool **bcscale** (int `$scale`)

Sets the default scale parameter for all subsequent bc math functions that do not explicitly specify a scale parameter.

Parameters

`scale`
The scale factor.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #7 - bcscale() **example**

```
<?php

// default scale : 3
bcscale(3);
echo bcdiv('105', '6.55957'); // 16.007

// this is the same without bcscale()
echo bcdiv('105', '6.55957', 3); // 16.007

?>
```

bcsqrt

bcsqrt -- Get the square root of an arbitrary precision number

Description

string **bcsqrt** (string `$operand` [, int `$scale`])

Return the square root of the *`operand`*.

Parameters

`operand`
: The operand, as a string.

`scale`
: This optional parameter is used to set the number of digits after the decimal place in the result. You can also set the global default scale for all functions by using bcscale().

Return Values

Returns the square root as a string, or **NULL** if *`operand`* is negative.

Examples

Example #8 - bcsqrt() example

```
<?php

echo bcsqrt('2', 3); // 1.414

?>
```

See Also

- bcpow()

bcsub

bcsub -- Subtract one arbitrary precision number from another

Description

string **bcsub** (string `$left_operand`, string `$right_operand` [, int `$scale`])

Subtracts the *`right_operand`* from the *`left_operand`*.

Parameters

`left_operand`
: The left operand, as a string.

`right_operand`
: The right operand, as a string.

`scale`
: This optional parameter is used to set the number of digits after the decimal place in the result. You can also set the global default scale for all functions by using bcscale().

Return Values

The result of the substraction, as a string.

Examples

Example #9 - bcsub() example

```
<?php

$a = '1.234';
$b = '5';

echo bcsub($a, $b);     // -3
echo bcsub($a, $b, 4);  // -3.7660

?>
```

See Also

- bcadd()