

## NAME

BitVector

## SYNOPSIS

```
use BitVector;

use BitVector ();

use BitVector qw(:all);
```

## DESCRIPTION

BitVector class provides the following methods:

new, ClearAllBits, ClearBit, ClearBits, ClearBitsRange, Copy, FlipAllBits, FlipBit, FlipBits, FlipBitsRange, GetBit, GetBitsAsBinaryString, GetBitsAsDecimalString, GetBitsAsHexadecimalString, GetBitsAsOctalString, GetBitsAsRawBinaryString, GetDensityOfClearBits, GetDensityOfSetBits, GetNumOfClearBits, GetNumOfSetBits, GetSize, IsBitClear, IsBitSet, IsBitVector, NewFromBinaryString, NewFromDecimalString, NewFromHexadecimalString, NewFromOctalString, NewFromRawBinaryString, Reverse, SetAllBits, SetBit, SetBitValue, SetBitValueBitOrder, SetBitValuePrintFormat, SetBits, SetBitsAsBinaryString, SetBitsAsDecimalString, SetBitsAsHexadecimalString, SetBitsAsOctalString, SetBitsAsRawBinaryString, SetBitsRange, StringifyBitVector

The following methods can also be used as functions:

IsBitVector, NewFromBinaryString, NewFromDecimalString, NewFromHexadecimalString, NewFromOctalString, NewFromRawBinaryString

The following operators are overloaded:

```
" " & | ^ ~ == !=
```

Things to keep in mind:

- o Bit numbers range from 0 to (Size - 1).
- o BitVector stored internally as bit in ascending order: LSB on the left
- o All bit data retrieval methods export data as LSB bit byte on the right: it makes bit data import and export consistent.
- o Stringify method provides an option to print LSB on the left.

## METHODS

new

```
$NewBitVector = new BitVector($Size);
```

Create a new *BitVector* object of size *Size* and return newly created BitVector. Bit numbers range from 0 to 1 less than *Size*

ClearAllBits

```
$BitVector->ClearAllBits();
```

Set all bit values to 0 in *BitVector* object and return *BitVector*

ClearBit

```
$BitVector->ClearBit($BitNum);
```

Set specified bit number *BitNum* to 0 in *BitVector* object and return *BitVector*

ClearBits

```
$BitVector->ClearBits(@BitNums);
```

Set specified bit numbers *BitNums* to 0 in *BitVector* object and return *BitVector*

ClearBitsRange

```
$BitVector->ClearBitsRange($MinBitNum, $MaxBitNum);
```

Set specified bit numbers between *MinBitNum* and *MaxBitNum* to 0 in *BitVector* object and return *BitVector*

Copy

```
$NewBitVector = $BitVector->Copy();
```

Copy *BitVector* and its associated data to a new BitVector and return a new BitVector

FlipAllBits

```
$BitVector->FlipAllBits();
```

Flip values of all bits in *BitVector* and its associated data to a new *BitVector* and return *BitVector*

#### FlipBit

```
$BitVector->FlipBit($BitNum);
```

Flip value of specified *BitNum* of in *BitVector* and return *BitVector*

#### FlipBits

```
$BitVector->FlipBits(@BitNums);
```

Flip values of specified bit numbers *BitNums* in *BitVector* object and return *BitVector*

#### FlipBitsRange

```
$BitVector->FlipBitsRange($MinBitNum, $MaxBitNum);
```

Flip values of specified bit numbers between *MinBitNum* and *MaxBitNum* in *BitVector* object and return *BitVector*

#### GetBit

```
$BitValue = $BitVector->GetBit($BitNum);
```

Returns value of bit number *BitNum* in *BitVector* object

#### GetBitsAsBinaryString

```
$BitString = $BitVector->GetBitsAsBinaryString();
```

Returns values of bits in *BitVector* as an ascii bit string containing 0s and 1s.

Internal bit values stored using Perl vec function with least significant bit and least significant byte on the left are converted into a binary string with least significant bit and least significant byte on the right.

#### GetBitsAsDecimalString

```
$BitString = $BitVector->GetBitsAsDecimalString();
```

Returns values of bits in *BitVector* as a decimal bit string containing values from 0 to 9.

Internal bit values stored using Perl vec function with least significant bit and least significant byte on the left are converted into a binary string with least significant bit and least significant byte on the right.

#### GetBitsAsHexadecimalString

```
$BitString = $BitVector->GetBitsAsHexadecimalString();
```

Returns values of bits in *BitVector* as a hexadecimal bit string containing values from 0 to 9 and a to f.

Internal bit values stored using Perl vec function with least significant bit and least significant byte on the left are converted into a binary string with least significant bit and least significant byte on the right.

#### GetBitsAsOctalString

```
$BitString = $BitVector->GetBitsAsOctalString();
```

Returns values of bits in *BitVector* as an octal bit string containing values from 0 to 7.

Internal bit values stored using Perl vec function with least significant bit and least significant byte on the left are converted into a binary string with least significant bit and least significant byte on the right.

#### GetBitsAsRawBinaryString

```
$BitString = $BitVector->GetBitsAsRawBinaryString();
```

Returns values of bits in *BitVector* as a string corresponding to packed bit values used by Perl vec function without performing any unpacking

#### GetDensityOfClearBits

```
$ClearBitsDensity = $BitVector->GetDensityOfClearBits();
```

Returns density of clear bits in *BitVector* which corresponds to number of bits set to 0 *BitVector* divided by its size

#### GetDensityOfSetBits

```
$SetBitsDensity = $BitVector->GetDensityOfSetBits();
```

Returns density of set bits in *BitVector* which corresponds to number of bits set to 1 in *BitVector* divided by its size

#### GetNumOfClearBits

```
$NumOfClearBits = $BitVector->GetNumOfClearBits();
```

Returns number of bits set to 0 in *BitVector*

#### GetNumOfSetBits

```
$NumOfSetBits = $BitVector->GetNumOfSetBits();
```

Returns number of bits set to 1 in *BitVector*

#### GetSize

```
$Size = $BitVector->GetSize();
```

Returns size of *BitVector*

#### IsBitClear

```
$Status = $BitVector->IsBitClear();
```

Returns 1 or 0 based on whether *BitNum* is set to 0 in *BitVector*

#### IsBitSet

```
$Status = $BitVector->IsBitSet($BitNum);
```

Returns 1 or 0 based on whether *BitNum* is set to 1 in *BitVector*

#### IsBitVector

```
$Status = BitVector::IsBitVector($Object);
```

Returns 1 or 0 based on whether *Object* is a BitVector object.

#### NewFromBinaryString

```
$NewBitVector = BitVector::NewFromBinaryString($BinaryString);
$NewBitVector = $BitVector->NewFromBinaryString($BinaryString);
```

Creates a new *BitVector* using *BinaryString* and returns new BitVector object

#### NewFromDecimalString

```
$NewBitVector = BitVector::NewFromDecimalString($DecimalString);
$NewBitVector = $BitVector->NewFromDecimalString($DecimalString);
```

Creates a new *BitVector* using *DecimalString* and returns new BitVector object

#### NewFromHexadecimalString

```
$NewBitVector = BitVector::NewFromHexadecimalString(
    $HexadecimalString);
$NewBitVector = $BitVector->NewFromHexadecimalString(
    $HexadecimalString);
```

Creates a new *BitVector* using *HexadecimalString* and returns new BitVector object

#### NewFromOctalString

```
$NewBitVector = BitVector::NewFromOctalString($OctalString);
$NewBitVector = $BitVector->NewFromOctalString($OctalString);
```

Creates a new *BitVector* using *OctalString* and returns new BitVector object

#### NewFromRawBinaryString

```
$NewBitVector = BitVector::NewFromRawBinaryString(
    $RawBinaryString);
$NewBitVector = $BitVector->NewFromRawBinaryString(
    $RawBinaryString);
```

Creates a new *BitVector* using *RawBinaryString* and returns new BitVector object

#### Reverse

```
$BitVector->Reverse();
```

Reverses values of bits in *BitVector* and returns *BitVector*. First bit number ends up with value of last bit number

#### SetAllBits

```
$BitVector->SetAllBits();
```

Sets values of all bits in *BitVector* to 1 and returns *BitVector*

#### SetBit

```
$BitVector->SetBit($BitNum);
```

Sets value of *BitNum* to 1 in *BitVector* and returns *BitVector*

#### SetBitValue

```
$BitVector->SetBitValue($BitNum, $BitValue);
```

Sets value of *BitNum* to *BitValue* in *BitVector* and returns *BitVector*

#### SetBitValueBitOrder

```
BitVector::SetBitValueBitOrder($BitOrder);
$BitVector->SetBitValueBitOrder($BitOrder);
```

Set bit order for printing BitVector values during stringification of BitVector object. Possible bit order values: *Ascending* or *Descending*.

Bit order can be set for either an individual BitVector object or the class.

Default is to print LSB bit in each byte on the right (*Descending* bit order). Internally, bits are stored in ascending order using Perl vec function and LSB bit in each byte on the left (*Ascending* bit order).

#### SetBitValuePrintFormat

```
BitVector::SetBitValuePrintFormat($PrintValueFormat);
$BitVector->SetBitValuePrintFormat($PrintValueFormat);
```

Set bit values print format for printing BitVector values during stringification of BitVector object. Possible print format values: *Binary*, *Bin*, *Hexadecimal*, *Hex*, *Decimal*, *Dec*, *Octal*, *Oct*, *RawBinary*, *RawBin*. Default: *Binary*.

Bit values print format can be set for either an individual BitVector object or the class.

#### SetBits

```
$BitVector->SetBits(@BitNums);
```

Set specified bit numbers *BitNums* to 1 in *BitVector* object and return *BitVector*

#### SetBitsAsBinaryString

```
$BitVector->SetBitsAsBinaryString($BinaryString);
```

Set bit values in *BitVector* using specified *BinaryString* and return *BitVector*. The size of *BitVector* is not changed.

#### SetBitsAsDecimalString

```
$BitVector->SetBitsAsDecimalString($DecimalString);
```

Set bit values in *BitVector* using specified *DecimalString* and return *BitVector*. The size of *BitVector* is not changed.

#### SetBitsAsHexadecimalString

```
$BitVector->SetBitsAsHexadecimalString($HexadecimalString);
```

Set bit values in *BitVector* using specified *HexadecimalString* and return *BitVector*. The size of *BitVector* is not changed.

#### SetBitsAsOctalString

```
$BitVector->SetBitsAsOctalString($OctalString);
```

Set bit values in *BitVector* using specified *OctalString* and return *BitVector*. The size of *BitVector* is not changed.

#### SetBitsAsRawBinaryString

```
$BitVector->SetBitsAsRawBinaryString($RawBinaryString);
```

Set bit values in *BitVector* using specified *RawBinaryString* and return *BitVector*. The size of *BitVector* is not changed.

#### SetBitsRange

```
$BitVector->SetBitsRange($MinBitNum, $MaxBitNum);
```

Set specified bit numbers between *MinBitNum* and *MaxBitNum* to 1 in *BitVector* object and return *BitVector*

#### StringifyBitVector

```
$String = $BitVector->StringifyBitVector();
```

Returns a string containing information about *BitVector* object

## AUTHOR

Manish Sud <msud@san.rr.com>

## SEE ALSO

Vector.pm

## COPYRIGHT

Copyright (C) 2004-2008 Manish Sud. All rights reserved.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.