NAME

TextUtil

SYNOPSIS

```
use TextUtil;

use TextUtil qw(:all);
```

DESCRIPTION

TextUtil module provides the following functions:

AddNumberSuffix, ContainsWhiteSpaces, GetTextFileDataByNonUniqueKey, GetTextFileDataByUniqueKey, GetTextLine, HashCode, IsEmpty, IsFloat, IsInteger, IsNotEmpty, IsNumberPowerOfNumber, IsNumerical, IsPositiveInteger, JoinWords, QuoteAWord, RemoveLeadingAndTrailingWhiteSpaces, RemoveLeadingWhiteSpaces, RemoveTrailingWhiteSpaces, SplitWords, WrapText

FUNCTIONS

AddNumberSuffix

```
$NumberWithSuffix = AddNumberSuffix($IntegerValue);
```

Returns number with appropriate suffix: 0, 1st, 2nd, 3rd, 4th, and so on.

ContainsWhiteSpaces

```
$Status = ContainsWhiteSpaces($TheString);
```

Returns 1 or 0 based on whether the string contains any white spaces.

GetTextLine

```
$Line = GetTextLine(\*TEXTFILE);
```

Reads next line from an already opened text file, takes out any carriage return, and returns it as a string. NULL is returned for EOF.

GetTextFileDataByNonUniqueKey

```
GetTextFileDataByNonUniqueKey($TextDataFile, $TextDataMapRef,
                              $DataKeyColNum, $InDelim);
```

Load data from a text file into the specified hash reference using a specific column for non-unique data key values.

The lines starting with # are treated as comments and ignored. First line not starting with # must contain column labels and the number of columns in all other data rows must match the number of column labels.

The first column is assumed to contain data key value by default; all other columns contain data as indicated in their column labels.

In order to avoid dependence of data access on the specified column labels, the column data is loaded into hash with Column<Num> hash keys, where column number start from 1. The data key column is not available as Colnum<Num> hash key;

The format of the data structure loaded into a specified hash reference is:

```
@{$TextDataMapRef->{DataKeys}} - Array of unique data keys
@{$TextDataMapRef->{ColLabels}} - Array of column labels
@{$TextDataMapRef->{DataColIDs}} - Array of data column IDs
$TextDataMapRef->{NumOfCols} - Number of columns
%{$TextDataMapRef->{DataKey}} - Hash keys pair: <DataKey, DataKey>
@{$TextDataMapRef->{DataCol<Num>}} - Hash keys pair with data as an array:
                                     <DataCol<Num>, DataKey>
```

GetTextFileDataByUniqueKey

```
GetTextFileDataByUniqueKey($TextDataFile, $TextDataMapRef, $DataKeyColNum,
                           $InDelim);
```

Load data from a text file into the specified hash reference using a a specific column for unique data key values.

The lines starting with # are treated as comments and ignored. First line not starting with # must contain column labels and the number of columns in all other data rows must match the number of column labels.

The first column is assumed to contain data key value by default; all other columns contain data as indicated in their column labels.

In order to avoid dependence of data access on the specified column labels, the column data is loaded into hash with Column<Num> hash keys, where column number start from 1. The data key column is not available as Colnum<Num> hash key;

The format of the data structure loaded into a specified hash reference is:

```
@{$TextDataMapRef->{DataKeys}} - Array of unique data keys
@{$TextDataMapRef->{ColLabels}} - Array of column labels
@{$TextDataMapRef->{DataColIDs}} - Array of data column IDs
$TextDataMapRef->{NumOfCols} - Number of columns
%{$TextDataMapRef->{DataKey}} - Hash keys pair: <DataKey, DataKey>
%{$TextDataMapRef->{DataCol<Num>}} - Hash keys pair: <DataCol<Num>, DataKey>
```

HashCode

```
$HashCode = HashCode($TheString);
```

Returns a 32 bit integer hash code using One-at-a-time algorithm By Bob Jenkins [Ref 38]. It's also implemented in Perl for internal hash keys in hv.h include file.

IsEmpty

```
$Status = IsEmpty($TheString);
```

Returns 1 or 0 based on whether the string is empty.

IsInteger

```
$Status = IsInteger($TheString);
```

Returns 1 or 0 based on whether the string is a positive integer.

IsPositiveInteger

```
$Status = IsPositiveInteger($TheString);
```

Returns 1 or 0 based on whether the string is an integer.

IsFloat

```
$Status = IsFloat($TheString);
```

Returns 1 or 0 based on whether the string is a float.

IsNotEmpty

```
$Status = IsNotEmpty($TheString);
```

Returns 0 or 1 based on whether the string is empty.

IsNumerical

```
$Status = IsNumerical($TheString);
```

Returns 1 or 0 based on whether the string is a number.

IsNumberPowerOfNumber

```
$Status = IsNumberPowerOfNumber($FirstNum, $SecondNum);
```

Returns 1 or 0 based on whether the first number is a power of second number.

JoinWords

```
$JoinedWords = JoinWords($Words, $Delim, $Quote);
```

Joins different words using delimiter and quote parameters, and returns it as a string.

QuoteAWord

```
$QuotedWord = QuoteAWord($Word, $Quote);
```

Returns a quoted string based on *Quote* value.

RemoveLeadingWhiteSpaces

```
$OutString = RemoveLeadingWhiteSpaces($InString);
```

Returns a string without any leading and traling white spaces.

RemoveTrailingWhiteSpaces

```
$OutString = RemoveTrailingWhiteSpaces($InString);
```

Returns a string without any trailing white spaces.

RemoveLeadingAndTrailingWhiteSpaces

```
$OutString = RemoveLeadingAndTrailingWhiteSpaces($InString);
```

Returns a string without any leading and traling white spaces.

SplitWords

```
@Words = SplitWords($Line, $Delimiter);
```

Returns an array *Words* ontaining unquoted words generated after spliting string value *Line* containing quoted or unquoted words.

This function is used to split strings generated by JoinWords as replacement for Perl's core module funtion Text::ParseWords::quotewords() which dumps core on very long strings.

WrapText

```
$OutString = WrapText($InString, [$WrapLength, $WrapDelimiter]);
```

Returns a wrapped string. By default, *WrapLenght* is *40* and *WrapDelimiter* is Unix new line character.

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

FileUtil.pm

COPYRIGHT