NAME

Tie::IxHash − ordered associative arrays for Perl

SYNOPSIS

```
    # simple usage
    use Tie::IxHash;
    tie HASHVARIABLE, 'Tie::IxHash' [, LIST];

    # OO interface with more powerful features
    use Tie::IxHash;
    TIEOBJECT = Tie::IxHash−>new( [LIST] );
    TIEOBJECT−>Splice( OFFSET [, LENGTH [, LIST]] );
    TIEOBJECT−>Push( LIST );
    TIEOBJECT−>Pop;
    TIEOBJECT−>Shift;
    TIEOBJECT−>Unshift( LIST );
    TIEOBJECT−>Keys( [LIST] );
    TIEOBJECT−>Values( [LIST] );
    TIEOBJECT−>Indices( LIST );
    TIEOBJECT−>Delete( [LIST] );
    TIEOBJECT−>Replace( OFFSET, VALUE, [KEY] );
    TIEOBJECT−>Reorder( LIST );
    TIEOBJECT−>SortByKey;
    TIEOBJECT−>SortByValue;
    TIEOBJECT−>Length;
```

DESCRIPTION

This Perl module implements Perl hashes that preserve the order in which the hash elements were added. The order is not affected when values corresponding to existing keys in the IxHash are changed. The elements can also be set to any arbitrary supplied order. The familiar perl array operations can also be performed on the IxHash.

Standard `TIEHASH` Interface

The standard `TIEHASH` mechanism is available. This interface is recommended for simple uses, since the usage is exactly the same as regular Perl hashes after the `tie` is declared.

Object Interface

This module also provides an extended object-oriented interface that can be used for more powerful operations with the IxHash. The following methods are available:

FETCH, STORE, DELETE, EXISTS

These standard `TIEHASH` methods mandated by Perl can be used directly. See the`tie` entry in **perlfunc** (1) for details.

Push, Pop, Shift, Unshift, Splice

These additional methods resembling Perl functions are available for operating on key-value pairs in the IxHash. The behavior is the same as the corresponding perl functions, except when a supplied hash key already exists in the hash. In that case, the existing value is updated but its order is not affected. To unconditionally alter the order of a supplied key-value pair, first `DELETE` the IxHash element.

Keys

Returns an array of IxHash element keys corresponding to the list of supplied indices. Returns an array of all the keys if called without arguments. Note the return value is mostly only useful when used in a list context (since perl will convert it to the number of elements in the array when used in a scalar context, and that may not be very useful).

If a single argument is given, returns the single key corresponding to the index. This is usable in either scalar or list context.

Values Returns an array of IxHash element values corresponding to the list of supplied indices. Returns an array of all the values if called without arguments. Note the return value is mostly only useful when used in a list context (since perl will convert it to the number of elements in the array when used in a scalar context, and that may not be very useful).

If a single argument is given, returns the single value corresponding to the index. This is usable in either scalar or list context.

Indices Returns an array of indices corresponding to the supplied list of keys. Note the return value is mostly only useful when used in a list context (since perl will convert it to the number of elements in the array when used in a scalar context, and that may not be very useful).

If a single argument is given, returns the single index corresponding to the key. This is usable in either scalar or list context.

Delete Removes elements with the supplied keys from the IxHash.

Replace Substitutes the IxHash element at the specified index with the supplied value-key pair. If a key is not supplied, simply substitutes the value at index with the supplied value. If an element with the supplied key already exists, it will be removed from the IxHash first.

Reorder This method can be used to manipulate the internal order of the IxHash elements by supplying a list of keys in the desired order. Note however, that any IxHash elements whose keys are not in the list will be removed from the IxHash.

Length Returns the number of IxHash elements.

SortByKey
Reorders the IxHash elements by textual comparison of the keys.

SortByValue
Reorders the IxHash elements by textual comparison of the values.

Clear Resets the IxHash to its pristine state: with no elements at all.

EXAMPLE

```
use Tie::IxHash;

# simple interface
$t = tie(%myhash, 'Tie::IxHash', 'a' => 1, 'b' => 2);
%myhash = (first => 1, second => 2, third => 3);
$myhash{fourth} = 4;
@keys = keys %myhash;
@values = values %myhash;
print("y") if exists $myhash{third};

# OO interface
$t = Tie::IxHash->new(first => 1, second => 2, third => 3);
$t->Push(fourth => 4); # same as $myhash{'fourth'} = 4;
($k, $v) = $t->Pop;    # $k is 'fourth', $v is 4
$t->Unshift(neg => -1, zeroth => 0);
($k, $v) = $t->Shift;  # $k is 'neg', $v is -1
@oneandtwo = $t->Splice(1, 2, foo => 100, bar => 101);

@keys = $t->Keys;
@values = $t->Values;
@indices = $t->Indices('foo', 'zeroth');
@itemkeys = $t->Keys(@indices);
@itemvals = $t->Values(@indices);
$t->Replace(2, 0.3, 'other');
$t->Delete('second', 'zeroth');
```

```
    $len = $t->Length;     # number of key-value pairs

    $t->Reorder(reverse @keys);
    $t->SortByKey;
    $t->SortByValue;
```

BUGS

You cannot specify a negative length to `Splice`. Negative indexes are OK, though.

NOTE

Indexing always begins at 0 (despite the current `$[` setting) for all the functions.

TODO

Addition of elements with keys that already exist to the end of the IxHash must be controlled by a switch.

Provide `TIEARRAY` interface when it stabilizes in Perl.

Rewrite using XSUBs for efficiency.

AUTHOR

Gurusamy Sarathy   gsar@umich.edu

VERSION

Version 1.23

SEE ALSO

perl(1)