ring

package | standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 0 | Imported by: 2,069

Details Valid go.mod file · Redistributable license · Tagged version · Stable version · Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Documentation

Overview

Package ring implements operations on circular lists.

Index

Examples

Constants

This section is empty.

Variables

This section is empty.

Functions

This section is empty.

Types

type Ring

```
type Ring struct {
    Value any // for use by client; untouched by this library
    // contains filtered or unexported fields
}
```

A Ring is an element of a circular list, or ring. Rings do not have a beginning or end; a pointer to any ring element serves as reference to the entire ring. Empty rings are represented as nil Ring pointers. The zero value for a Ring is a one-element ring with a nil Value.

func New

```
func New(n int) *Ring
```

New creates a ring of n elements.

func (*Ring) Do

```
func (r *Ring) Do(f func(any))
```

Do calls function f on each element of the ring, in forward order. The behavior of Do is undefined if f changes *r.

▶ Example

func (*Ring) Len

```
func (r *Ring) Len() int
```

Len computes the number of elements in ring r. It executes in time proportional to the number of elements.

▶ Example

func (*Ring) Link

```
func (r *Ring) Link(s *Ring) *Ring
```

Link connects ring r with ring s such that r.Next() becomes s and returns the original value for r.Next(). r must not be empty.

If r and s point to the same ring, linking them removes the elements between r and s from the ring. The removed elements form a subring and the result is a reference to that subring (if no elements were removed, the result is still the original value for r.Next(), and not nil).

If r and s point to different rings, linking them creates a single ring with the elements of s inserted after r. The result points to the element following the last element of s after insertion.

▶ Example

func (*Ring) Move

```
func (r *Ring) Move(n int) *Ring
```

Move moves n % r.Len() elements backward (n < 0) or forward (n >= 0) in the ring and returns that ring element. r must not be empty.

▶ Example

func (*Ring) Next

```
func (r *Ring) Next() *Ring
```

Next returns the next ring element. r must not be empty.

▶ Example

func (*Ring) Prev

```
func (r *Ring) Prev() *Ring
```

Prev returns the previous ring element. r must not be empty.

▶ Example

func (*Ring) Unlink

```
func (r *Ring) Unlink(n int) *Ring
```

Unlink removes n % r.Len() elements from the ring r, starting at r.Next(). If n % r.Len() == 0, r remains unchanged. The result is the removed subring. r must not be empty.

▶ Example

Source Files

View all

ring.go