**Github**[**https://github.com/Angelrggarcia/TC3002B.505/tree/main**](https://github.com/Angelrggarcia/TC3002B.505/tree/main) **1. Stack (Pila - LIFO)**

La pila se implementa usando un slice genérico:

* **Push**: Añade un elemento al final del slice (parte superior de la pila)
* **Pop**: Elimina y devuelve el último elemento del slice
* **Peek**: Devuelve (sin eliminar) el último elemento
* **IsEmpty**: Verifica si el slice está vacío
* **Size**: Devuelve la longitud del slice

**2. Queue (Cola - FIFO)**

La cola también usa un slice genérico:

* **Enqueue**: Añade un elemento al final del slice
* **Dequeue**: Elimina y devuelve el primer elemento del slice
* **Peek**: Devuelve (sin eliminar) el primer elemento
* **IsEmpty**: Verifica si el slice está vacío
* **Size**: Devuelve la longitud del slice

**3. Map (Diccionario)**

Para el diccionario usamos el tipo map nativo de Go que:

* Permite almacenar pares clave-valor
* Proporciona operaciones eficientes de inserción, acceso y eliminación
* Soporta verificación de existencia de claves

**Casos de prueba incluidos**

Los tests verifican:

1. **Stack**:
   * Comportamiento LIFO (el último en entrar es el primero en salir)
   * Operaciones con pila vacía
   * Correcto funcionamiento de Peek y Size
2. **Queue**:
   * Comportamiento FIFO (el primero en entrar es el primero en salir)
   * Operaciones con cola vacía
   * Correcto funcionamiento de Peek y Size
3. **Map**:
   * Inserción y recuperación de valores
   * Eliminación de elementos
   * Verificación de existencia de claves

**1. Pruebas para Stack (Pila)**

**Caso 1: Pila vacía**

pila := NewStack[int]()

fmt.Println(pila.IsEmpty()) // true

fmt.Println(pila.Size()) // 0

\_, ok := pila.Pop()

fmt.Println(ok) // false

\_, ok = pila.Peek()

fmt.Println(ok) // false  
  
**Resultados**  
true

0

false

false  
  
**Caso 2: Operaciones básicas**  
  
pila := NewStack[int]()

pila.Push(10)

pila.Push(20)

fmt.Println(pila.Size()) // 2

fmt.Println(pila.IsEmpty()) // false

val, \_ := pila.Peek()

fmt.Println(val) // 20

val, \_ = pila.Pop()

fmt.Println(val) // 20

val, \_ = pila.Pop()

fmt.Println(val) // 10

fmt.Println(pila.IsEmpty()) // true  
  
**Resultado**  
2

false

20

20

10

True  
  
**Pruebas para Queue (Cola)**

**Caso 1: Cola vacía**

cola := NewQueue[string]()

fmt.Println(cola.IsEmpty()) // true

fmt.Println(cola.Size()) // 0

\_, ok := cola.Dequeue()

fmt.Println(ok) // false

\_, ok = cola.Peek()

fmt.Println(ok) // false  
  
**Resultados**true

0

false

false  
  
**Caso 2: Operaciones básicas**

cola := NewQueue[float64]()

cola.Enqueue(3.14)

cola.Enqueue(2.71)

fmt.Println(cola.Size()) // 2

fmt.Println(cola.IsEmpty()) // false

val, \_ := cola.Peek()

fmt.Println(val) // 3.14

val, \_ = cola.Dequeue()

fmt.Println(val) // 3.14

val, \_ = cola.Dequeue()

fmt.Println(val) // 2.71

fmt.Println(cola.IsEmpty()) // true

**Resultado**2

false

3.14

3.14

2.71

True

**3. Pruebas para Map (Diccionario)**

**Caso 1: Diccionario vacío**

dic := make(map[int]string)

fmt.Println(len(dic)) // 0

\_, ok := dic[1]

fmt.Println(ok) // false  
**Resultados**0

false  
  
**Caso 2: Operaciones básicas**

dic := make(map[string]int)

dic["uno"] = 1

dic["dos"] = 2

fmt.Println(len(dic)) // 2

fmt.Println(dic["uno"]) // 1

val, ok := dic["tres"]

fmt.Println(val, ok) // 0 false

delete(dic, "uno")

fmt.Println(len(dic)) // 1

\_, ok = dic["uno"]

fmt.Println(ok) // false  
  
**Resultados**2

1

0 false

1

false