Arrays en Go

Un array es una colección de elementos del mismo tipo que tiene un tamaño fijo. Esto significa que una vez que se declara un array con un tamaño específico, no se puede cambiar.

Declaración y uso básico de arrays:

```go
package main

import "fmt"

func main() {
    // Declarar un array de 5 enteros
    var arr [5]int

    // Asignar valores
    arr[0] = 10
    arr[1] = 20
    arr[2] = 30
    // Imprimir el array
    fmt.Println("Array:", arr)
    // Acceder a un elemento
    fmt.Println("Elemento en la posición 1:", arr[1])
}
```

En este ejemplo, el array arr tiene un tamaño fijo de 5 elementos.
Los elementos no inicializados se establecen en el valor cero de su tipo (en este caso, 0 para los enteros).
Funciones útiles para arrays:
len(arr): Devuelve el tamaño (número de elementos) del array.

Slices en Go

Un slice es una vista dinámica sobre un array subyacente. A diferencia de los arrays, los slices no tienen un tamaño fijo; pueden crecer o reducirse según sea necesario.

Declaración y uso básico de slices:

```go
package main

import "fmt"

func main() {
    // Declarar un slice de enteros
    var slice []int

    // Agregar elementos usando append
    slice = append(slice, 10)
    slice = append(slice, 20, 30)

    // Imprimir el slice
    fmt.Println("Slice:", slice)

    // Acceder a un elemento
    fmt.Println("Elemento en la posición 1:", slice[1])
}
```

En este ejemplo, el slice slice comienza vacío, pero se pueden agregar elementos dinámicamente usando la función append.

Funciones útiles para slices:

- **len(slice)**: Devuelve el número de elementos del slice.
- **cap(slice)**: Devuelve la capacidad del slice (la capacidad del array subyacente).
- **append(slice, elementos)**: Agrega uno o más elementos al final del slice.
- **copy(destSlice, srcSlice)**: Copia elementos de un slice a otro.

Sub-slices:

Puedes crear un sub-slice de un slice o array existente usando la notación [inicio:fin].

```
package main

import "fmt"

func main() {
    // Declarar y inicializar un array
    arr := [5]int{10, 20, 30, 40, 50}
    // Crear un sub-slice del array
    subSlice := arr[1:4]
    // Imprimir el sub-slice
    fmt.Println("Sub-slice:", subSlice)
}
```

En este ejemplo, subSlice es un slice que contiene los elementos del índice 1 al 3 (no incluye el índice 4).

Resumen de diferencias entre arrays y slices:

Característica	Array	Slice
Tamaño	Fijo	Dinámico
Capacidad	Igual al tamaño	Puede crecer o reducirse
Inicialización	Se debe especificar tamaño	Puede ser vacío al inicio
Métodos útiles	len()	len(), cap(), append(), copy()