Day 22 - Go (Golang) Basics -Arrays

An array is a collection of similar data elements stored at contiguous memory locations.

- In Go, arrays are **homogeneous** i.e., all elements must be of the same data type.
- Arrays in Go have **fixed length** i.e., once declared, their size cannot change.
- The **length** of an array is the number of elements it holds.
- The **capacity** is the total number it *can* hold. For arrays, length == capacity.

Declaration Syntax:

`var <arrayName> [size] <dataType>`

Example:

```go
package main

import "fmt"

func main() {
    var fruits [5]string
    fmt.Println(fruits) // Output: [     ]
}


```

Initialization Examples:

Method 1 – Explicit declaration:

`var fruits [3]string = [3]string{"apple", "orange", "banana"}`

Method 2 – Shorthand:

`fruits := [3]string{"apple", "orange", "banana"}`

Method 3 – Using ellipses to infer length:

`fruits := [...]string{"apple", "orange", "banana"}`

Get Array Length:

Use `len()` function:

```go
package main

import "fmt"

func main() {
    var fruits [2]string = [2]string{"apple", "banana"}
    fmt.Println(len(fruits)) // Output: 2
}
```

Access and Modify Elements:

Elements are accessed by their index (starting from 0):

`fruits[0] = "mango"`

`fmt.Println(fruits[0])`

Looping Through Arrays:

Using a `for` loop:

```
for i := 0; i < len(fruits); i++ {
    fmt.Println(fruits[i])
}
```

Using `range`:

```go
package main

import "fmt"

func main() {
    fruits := [3]string{"apple", "orange", "banana"}

    for index, element := range fruits {
        fmt.Println(index, "->", element)
    }

}
```

Multidimensional Arrays:

Arrays within arrays are declared like `[rows][cols]datatype`

Example:

```go
package main

import "fmt"

func main() {
    food := [3][2]string{
        {"apple", "fruit"},
        {"potato", "vegetable"},
        {"rice", "grain"},
    }

   fmt.Println(food[1][1]) // Output: vegetable

}
```