PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU
FACULTAD DE CIENCIAS E INGENIERIA
INGENIERIA INFORMATICA

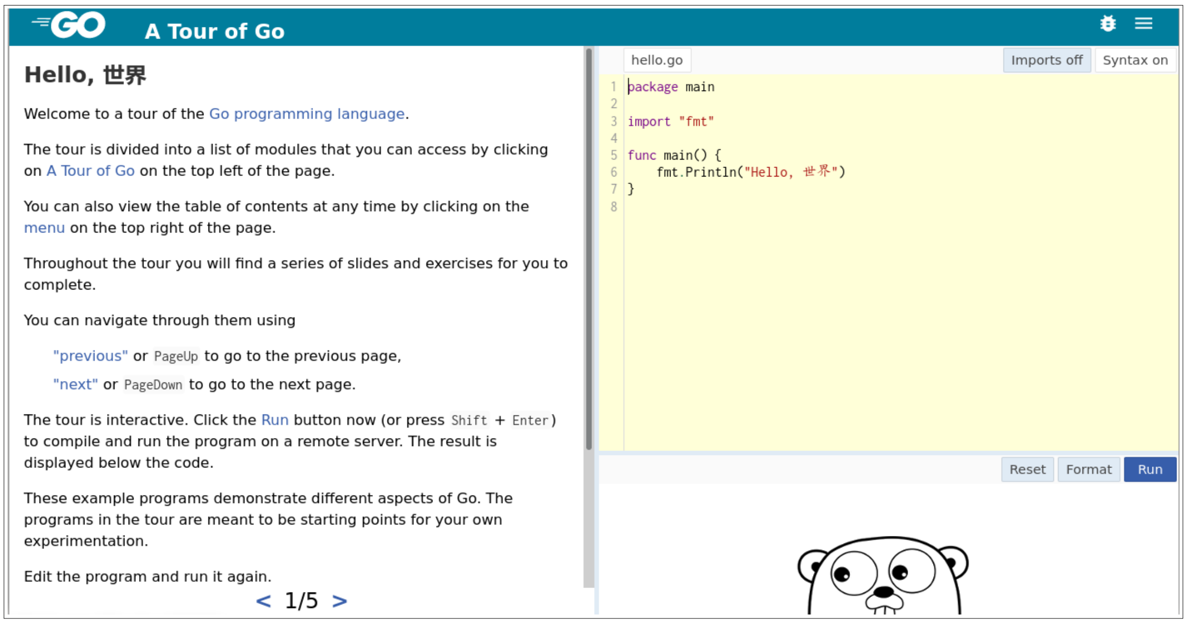
INF239 SISTEMAS OPERATIVOS
(Laboratorio)

Laboratorio Preliminar 0C

Lenguaje de Programación Go

Usted puede realizar un recorrido rápido sobre el lenguaje de programación Go, visitando:

https://go.dev/tour/welcome/1



Hello, 世界

Welcome to a tour of the Go programming language.

The tour is divided into a list of modules that you can access by clicking on A Tour of Go on the top left of the page.

You can also view the table of contents at any time by clicking on the menu on the top right of the page.

Throughout the tour you will find a series of slides and exercises for you to complete.

You can navigate through them using

"previous" or `PageUp` to go to the previous page,

"next" or `PageDown` to go to the next page.

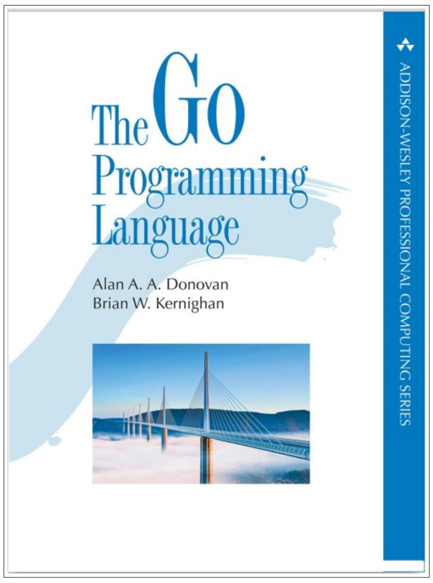
The tour is interactive. Click the Run button now (or press `Shift + Enter`) to compile and run the program on a remote server. The result is displayed below the code.

These example programs demonstrate different aspects of Go. The programs in the tour are meant to be starting points for your own experimentation.

Edit the program and run it again.

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello, 世界")
}
```

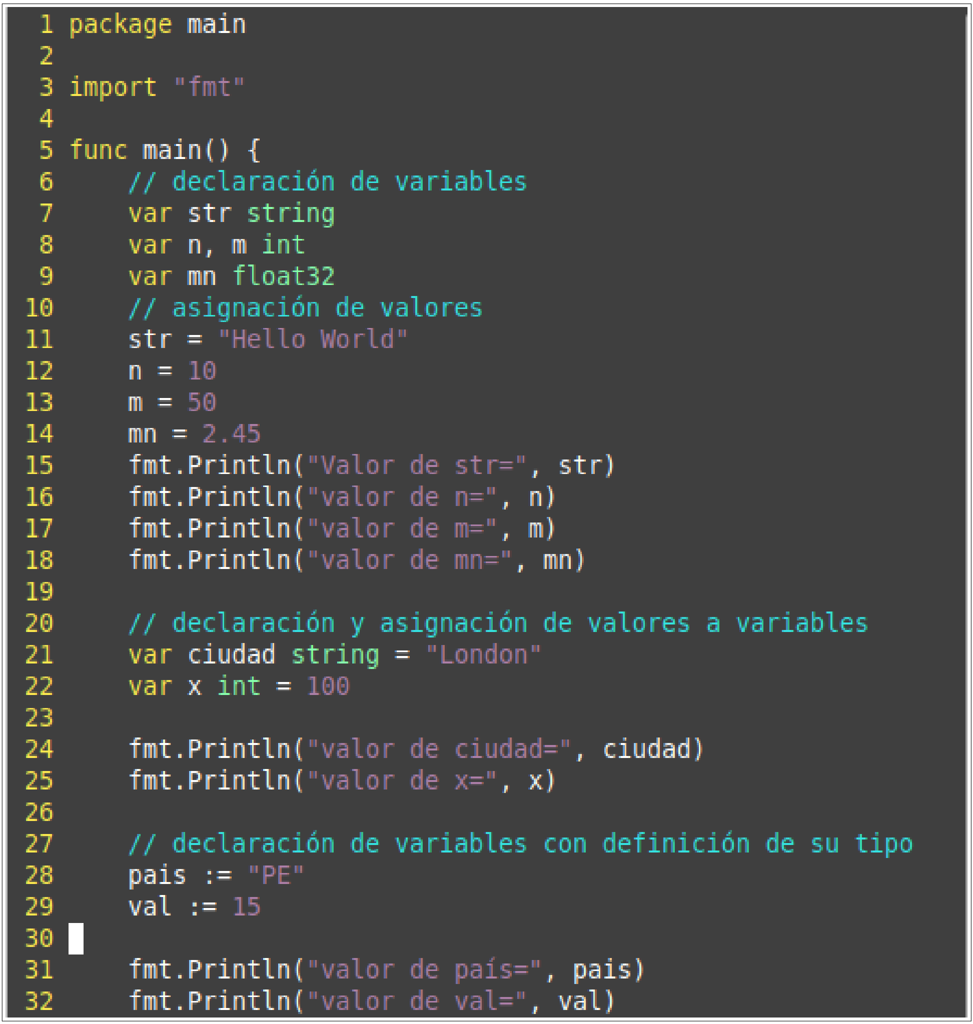
También puede consultar el siguiente libro:


The Go Programming Language
Alan A. A. Donovan
Brian W. Kernighan
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

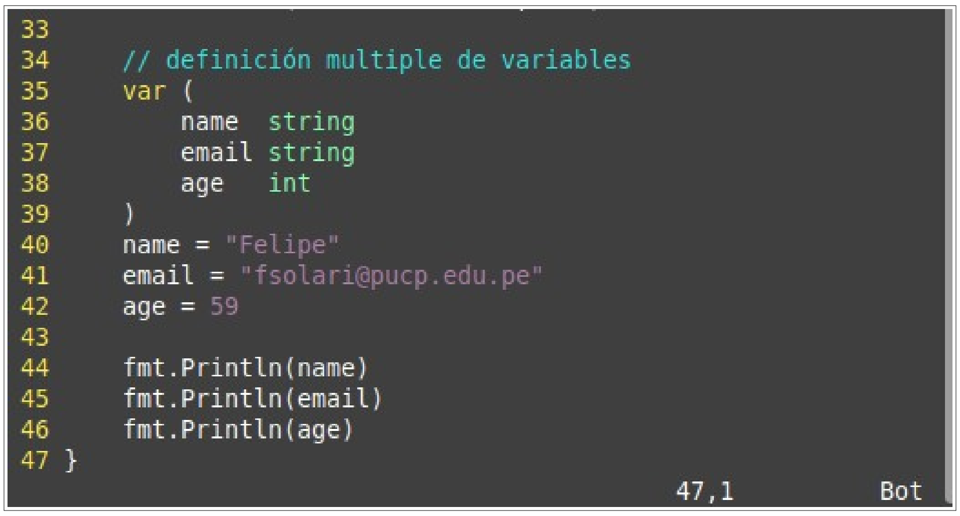
Tipos de datos básicos

```
bool

string

int  int8  int16  int32  int64
uint uint8 uint16 uint32 uint64 uintptr

byte // alias for uint8

rune // alias for int32
     // represents a Unicode code point

float32 float64

complex64 complex128
```

Declaración de variables

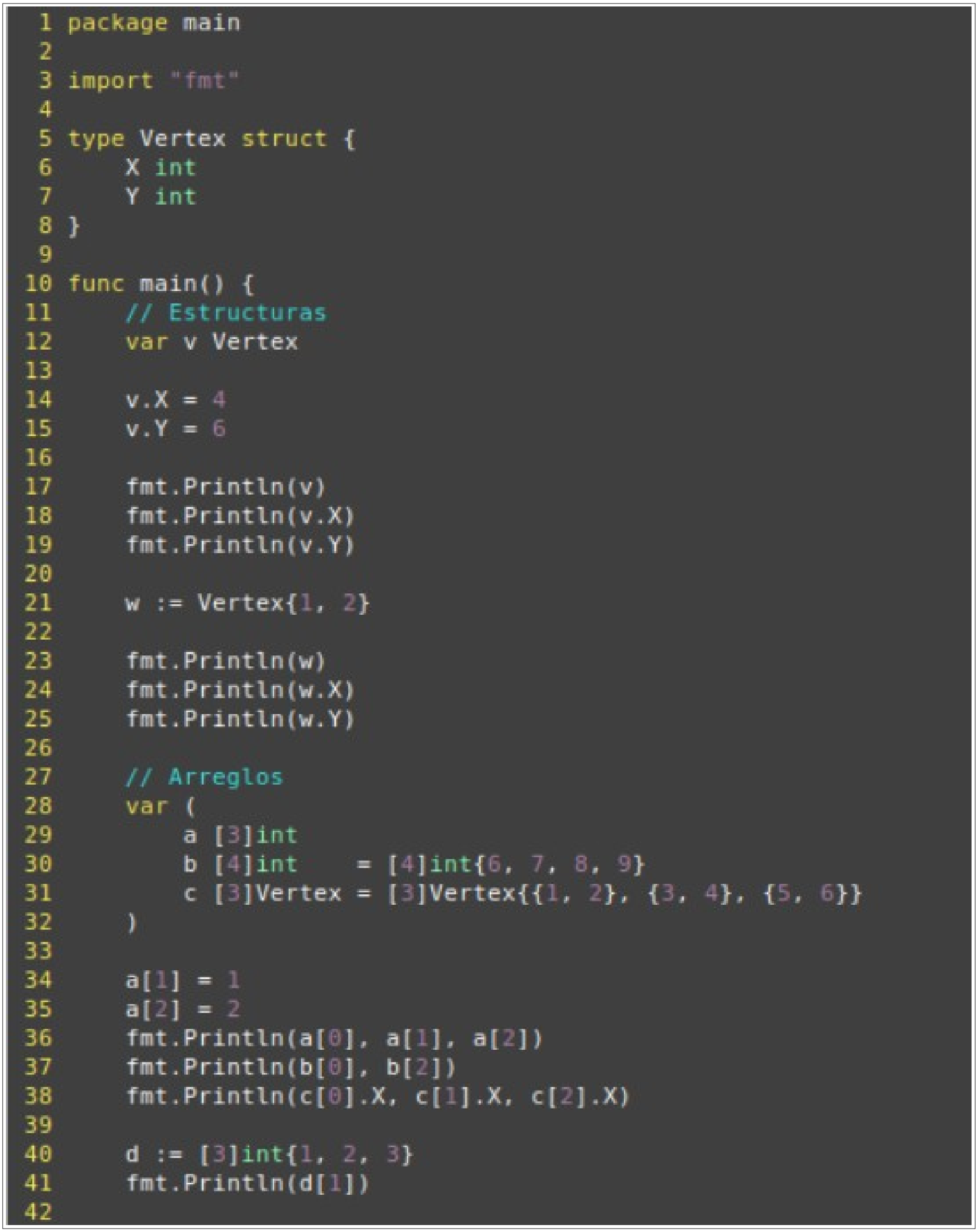


```go
package main

import "fmt"

func main() {
    // declaración de variables
    var str string
    var n, m int
    var mn float32
    // asignación de valores
    str = "Hello World"
    n = 10
    m = 50
    mn = 2.45
    fmt.Println("Valor de str=", str)
    fmt.Println("valor de n=", n)
    fmt.Println("valor de m=", m)
    fmt.Println("valor de mn=", mn)

    // declaración y asignación de valores a variables
    var ciudad string = "London"
    var x int = 100

    fmt.Println("valor de ciudad=", ciudad)
    fmt.Println("valor de x=", x)

    // declaración de variables con definición de su tipo
    pais := "PE"
    val := 15

    fmt.Println("valor de país=", pais)
    fmt.Println("valor de val=", val)
```

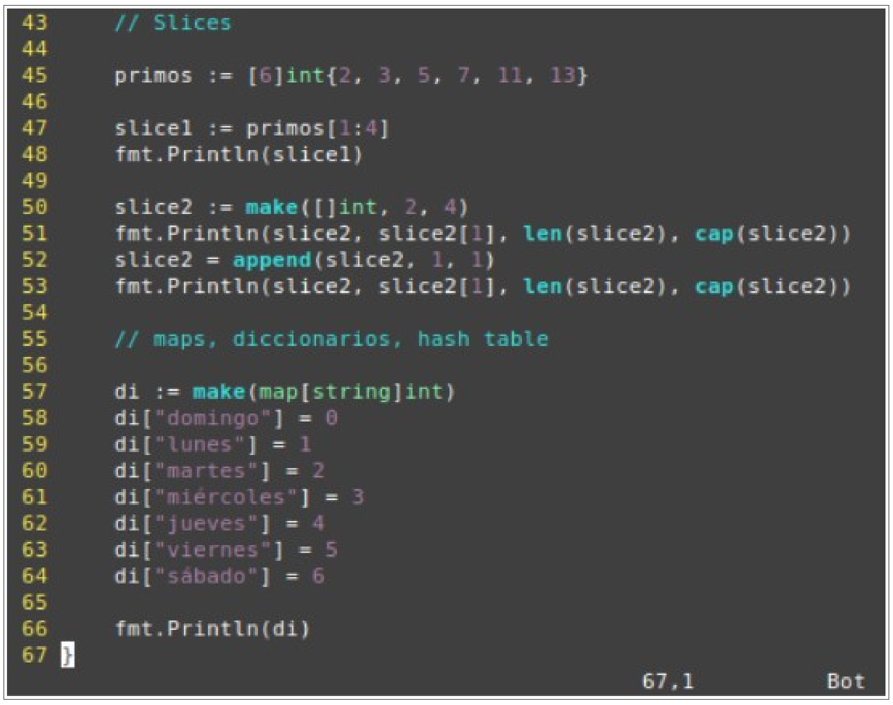


```

    // definición multiple de variables
    var (
        name  string
        email string
        age   int
    )
    name = "Felipe"
    email = "fsolari@pucp.edu.pe"
    age = 59

    fmt.Println(name)
    fmt.Println(email)
    fmt.Println(age)
}
                                                47,1          Bot
```

Estructuras, Arreglos, Slices y Maps

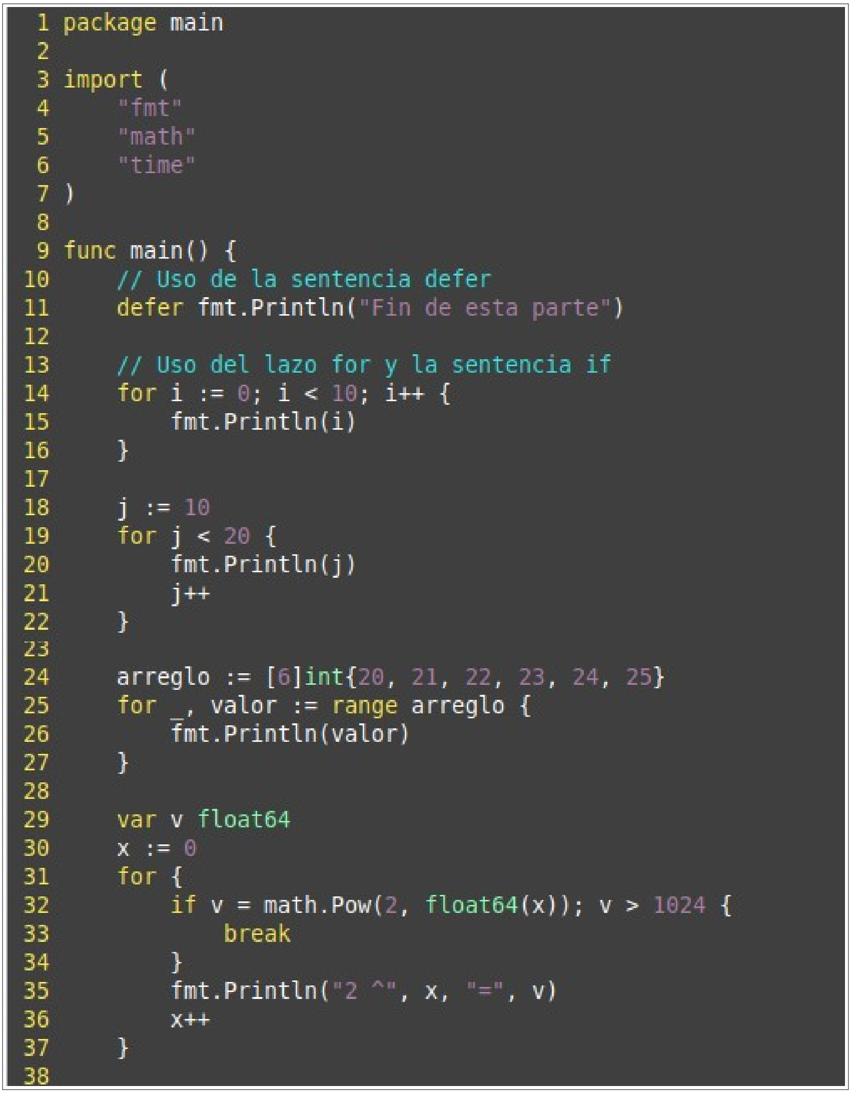


```go
package main

import "fmt"

type Vertex struct {
    X int
    Y int
}

func main() {
    // Estructuras
    var v Vertex

    v.X = 4
    v.Y = 6

    fmt.Println(v)
    fmt.Println(v.X)
    fmt.Println(v.Y)

    w := Vertex{1, 2}

    fmt.Println(w)
    fmt.Println(w.X)
    fmt.Println(w.Y)

    // Arreglos
    var (
        a [3]int
        b [4]int     = [4]int{6, 7, 8, 9}
        c [3]Vertex = [3]Vertex{{1, 2}, {3, 4}, {5, 6}}
    )

    a[1] = 1
    a[2] = 2
    fmt.Println(a[0], a[1], a[2])
    fmt.Println(b[0], b[2])
    fmt.Println(c[0].X, c[1].X, c[2].X)

    d := [3]int{1, 2, 3}
    fmt.Println(d[1])

```

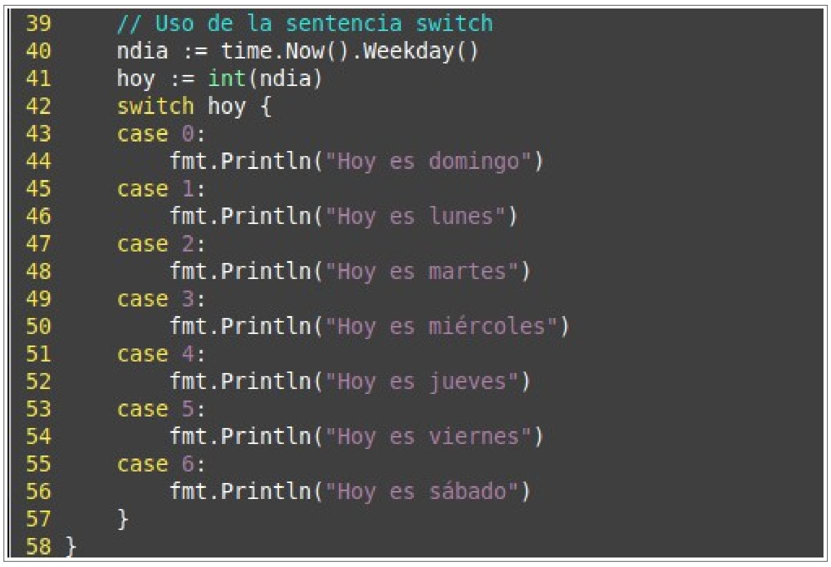


```go
    // Slices

    primos := [6]int{2, 3, 5, 7, 11, 13}

    slice1 := primos[1:4]
    fmt.Println(slice1)

    slice2 := make([]int, 2, 4)
    fmt.Println(slice2, slice2[1], len(slice2), cap(slice2))
    slice2 = append(slice2, 1, 1)
    fmt.Println(slice2, slice2[1], len(slice2), cap(slice2))

    // maps, diccionarios, hash table

    di := make(map[string]int)
    di["domingo"] = 0
    di["lunes"] = 1
    di["martes"] = 2
    di["miércoles"] = 3
    di["jueves"] = 4
    di["viernes"] = 5
    di["sábado"] = 6

    fmt.Println(di)
}
                                                  67,1          Bot
```

Sentencias de control de flujo: `for`, `if`, `switch`, `defer`

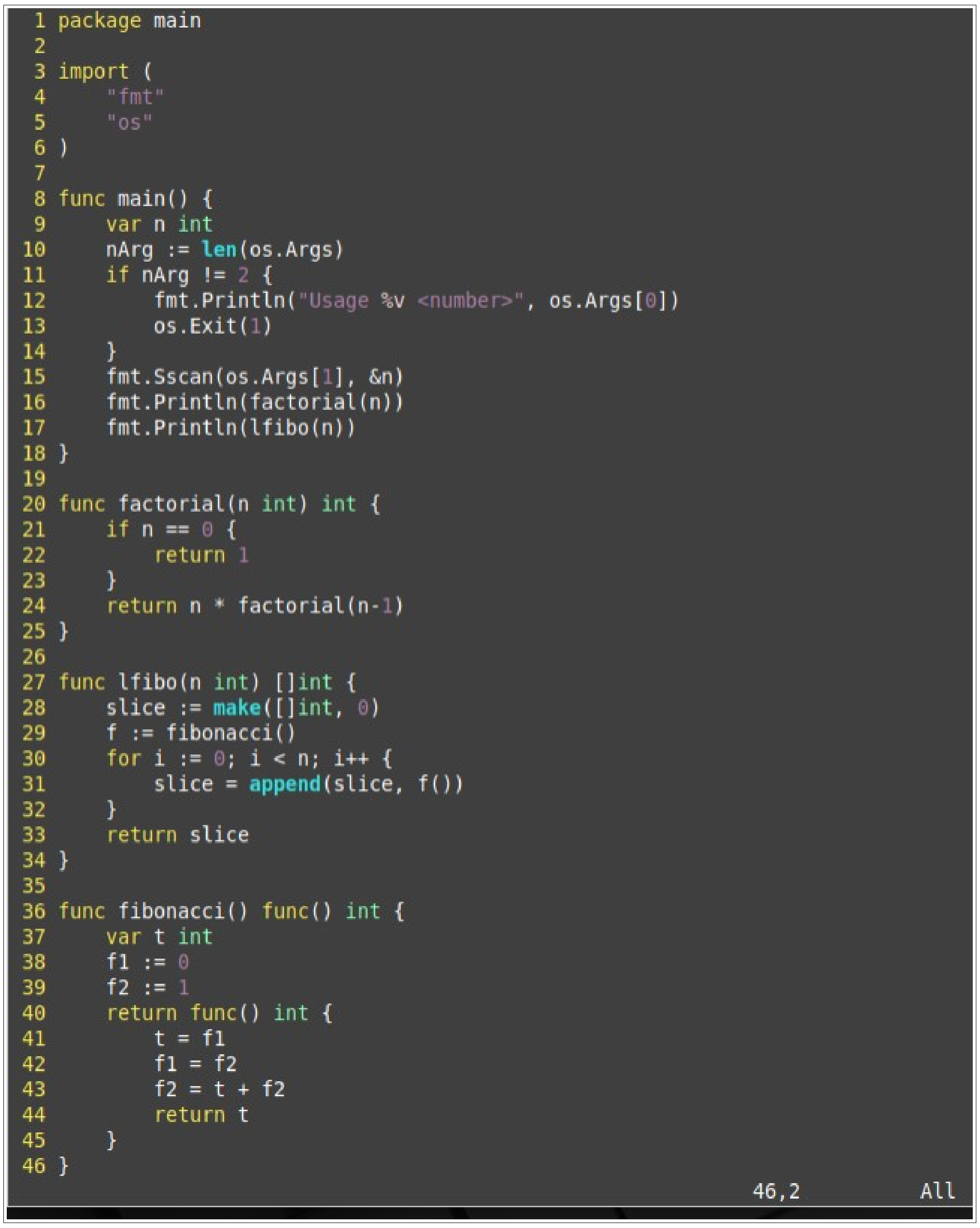


```go
package main

import (
    "fmt"
    "math"
    "time"
)

func main() {
    // Uso de la sentencia defer
    defer fmt.Println("Fin de esta parte")

    // Uso del lazo for y la sentencia if
    for i := 0; i < 10; i++ {
        fmt.Println(i)
    }

    j := 10
    for j < 20 {
        fmt.Println(j)
        j++
    }

    arreglo := [6]int{20, 21, 22, 23, 24, 25}
    for _, valor := range arreglo {
        fmt.Println(valor)
    }

    var v float64
    x := 0
    for {
        if v = math.Pow(2, float64(x)); v > 1024 {
            break
        }
        fmt.Println("2 ^", x, "=", v)
        x++
    }

```



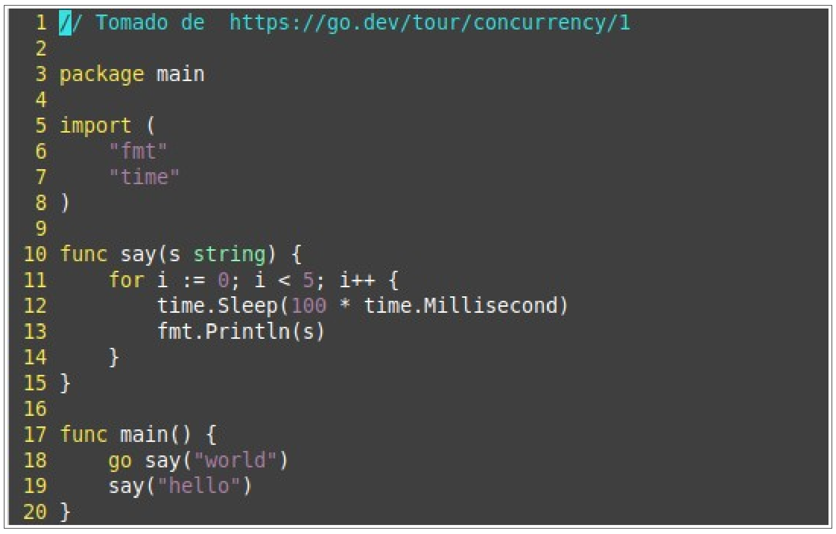
```go
    // Uso de la sentencia switch
    ndia := time.Now().Weekday()
    hoy := int(ndia)
    switch hoy {
    case 0:
        fmt.Println("Hoy es domingo")
    case 1:
        fmt.Println("Hoy es lunes")
    case 2:
        fmt.Println("Hoy es martes")
    case 3:
        fmt.Println("Hoy es miércoles")
    case 4:
        fmt.Println("Hoy es jueves")
    case 5:
        fmt.Println("Hoy es viernes")
    case 6:
        fmt.Println("Hoy es sábado")
    }
}
```

Funciones



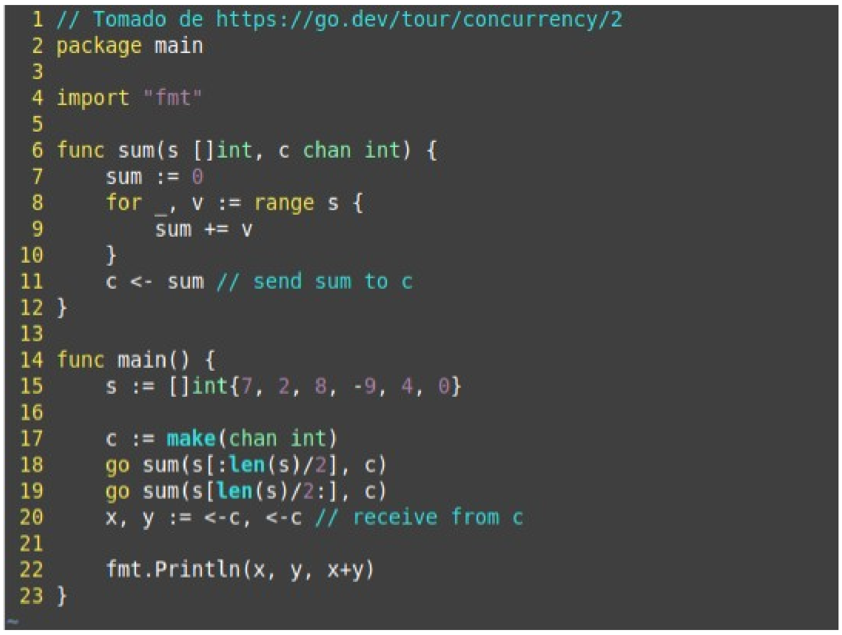
```go
package main

import (
    "fmt"
    "os"
)

func main() {
    var n int
    nArg := len(os.Args)
    if nArg != 2 {
        fmt.Println("Usage %v <number>", os.Args[0])
        os.Exit(1)
    }
    fmt.Sscan(os.Args[1], &n)
    fmt.Println(factorial(n))
    fmt.Println(lfibo(n))
}

func factorial(n int) int {
    if n == 0 {
        return 1
    }
    return n * factorial(n-1)
}

func lfibo(n int) []int {
    slice := make([]int, 0)
    f := fibonacci()
    for i := 0; i < n; i++ {
        slice = append(slice, f())
    }
    return slice
}

func fibonacci() func() int {
    var t int
    f1 := 0
    f2 := 1
    return func() int {
        t = f1
        f1 = f2
        f2 = t + f2
        return t
    }
}
```

Goroutines



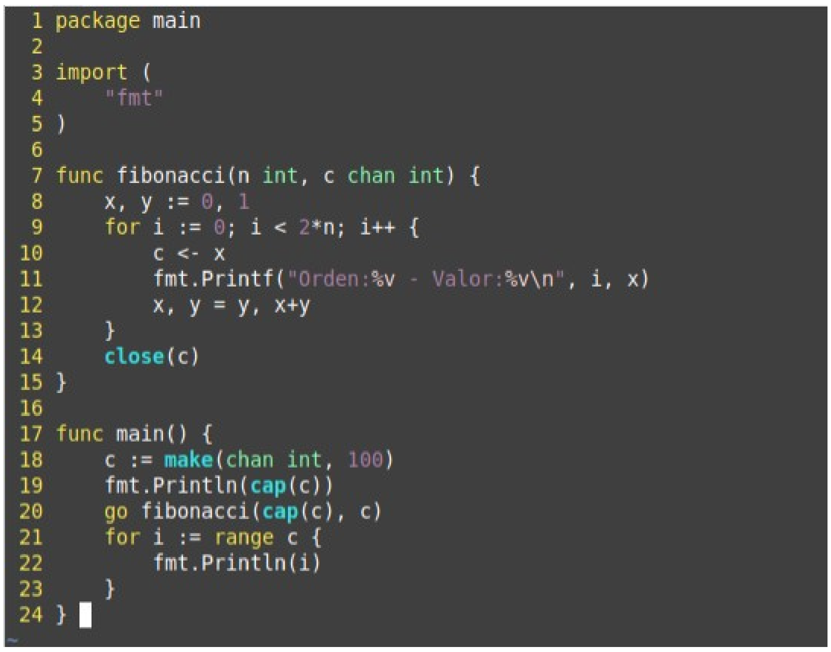
```go
// Tomado de  https://go.dev/tour/concurrency/1

package main

import (
    "fmt"
    "time"
)

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Println(s)
    }
}

func main() {
    go say("world")
    say("hello")
}
```

Channels



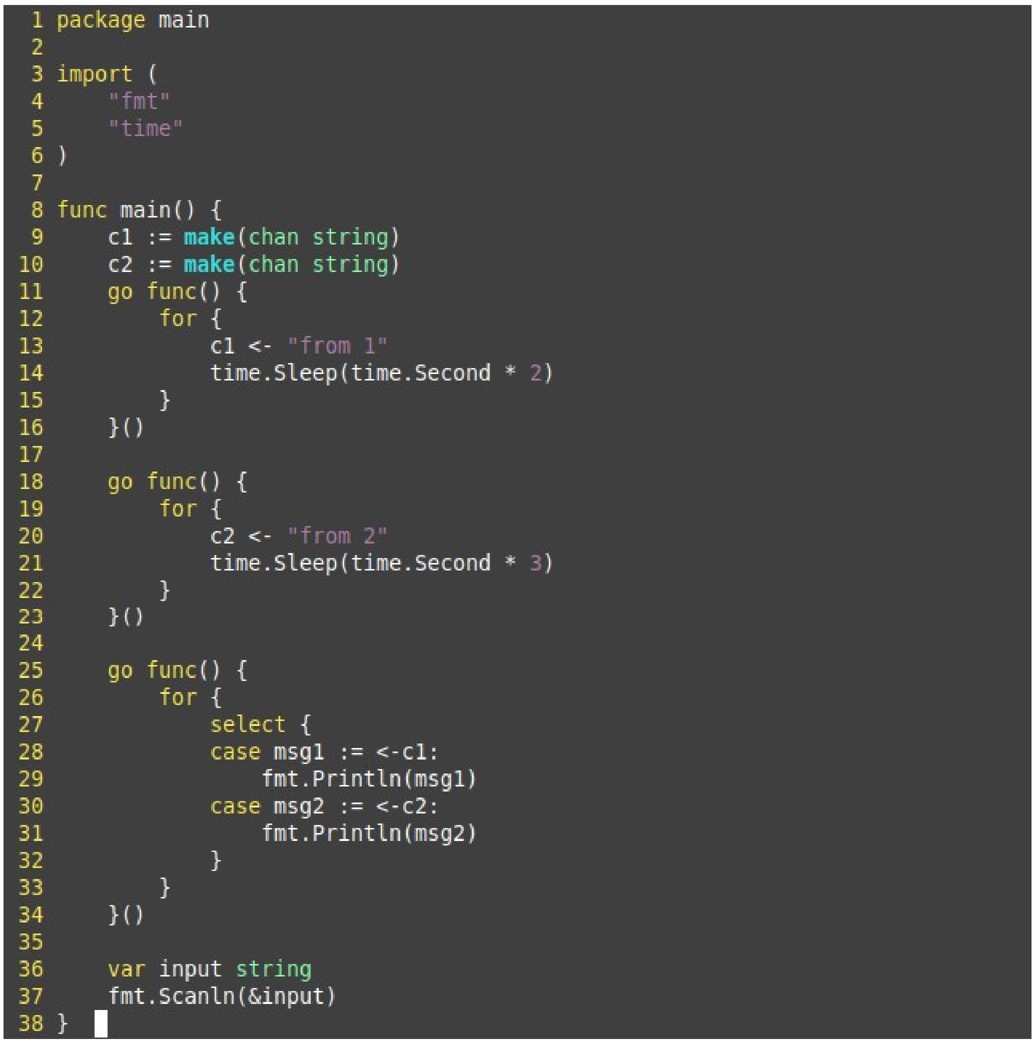
```go
// Tomado de https://go.dev/tour/concurrency/2
package main

import "fmt"

func sum(s []int, c chan int) {
    sum := 0
    for _, v := range s {
        sum += v
    }
    c <- sum // send sum to c
}

func main() {
    s := []int{7, 2, 8, -9, 4, 0}

    c := make(chan int)
    go sum(s[:len(s)/2], c)
    go sum(s[len(s)/2:], c)
    x, y := <-c, <-c // receive from c

    fmt.Println(x, y, x+y)
}
```

Buffered Channels



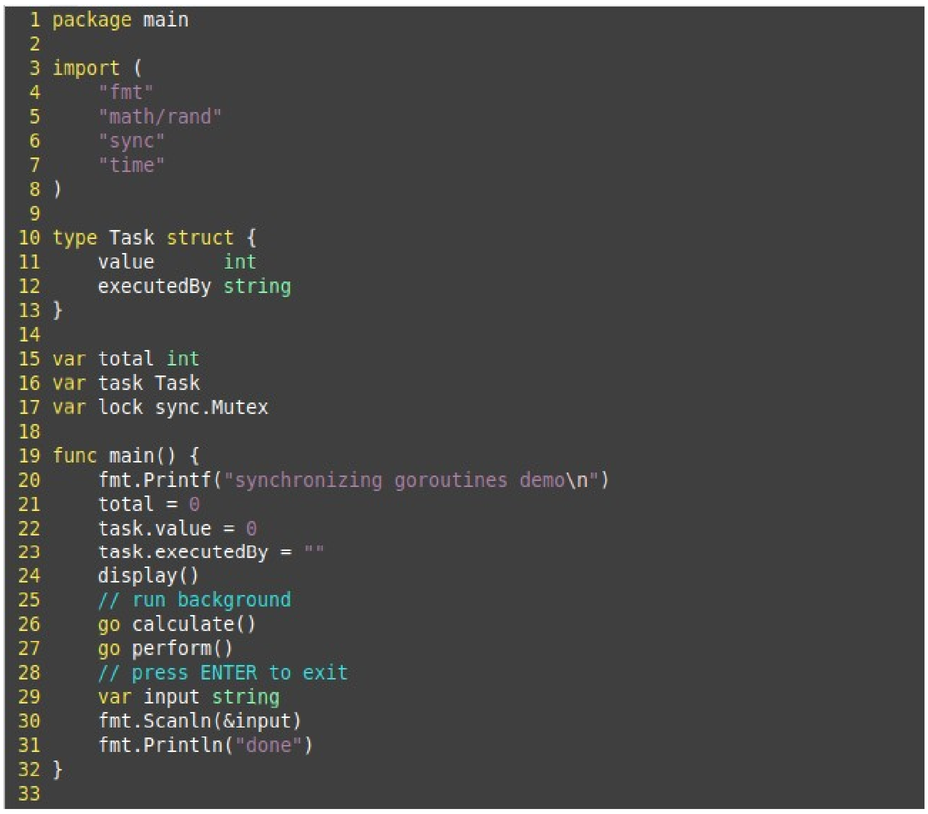
```go
package main

import (
    "fmt"
)

func fibonacci(n int, c chan int) {
    x, y := 0, 1
    for i := 0; i < 2*n; i++ {
        c <- x
        fmt.Printf("Orden:%v - Valor:%v\n", i, x)
        x, y = y, x+y
    }
    close(c)
}

func main() {
    c := make(chan int, 100)
    fmt.Println(cap(c))
    go fibonacci(cap(c), c)
    for i := range c {
        fmt.Println(i)
    }
}
```

Select

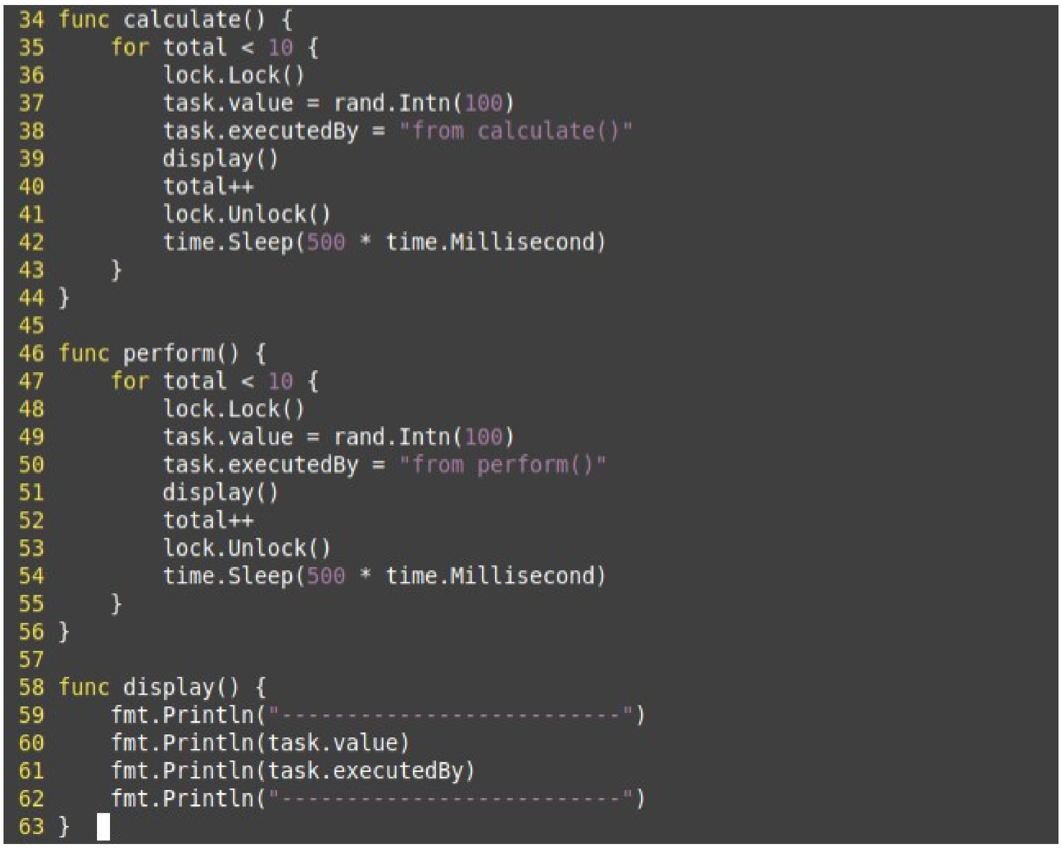


```go
package main

import (
    "fmt"
    "time"
)

func main() {
    c1 := make(chan string)
    c2 := make(chan string)
    go func() {
        for {
            c1 <- "from 1"
            time.Sleep(time.Second * 2)
        }
    }()

    go func() {
        for {
            c2 <- "from 2"
            time.Sleep(time.Second * 3)
        }
    }()

    go func() {
        for {
            select {
            case msg1 := <-c1:
                fmt.Println(msg1)
            case msg2 := <-c2:
                fmt.Println(msg2)
            }
        }
    }()

    var input string
    fmt.Scanln(&input)
}
```

Sincronización



```go
package main

import (
    "fmt"
    "math/rand"
    "sync"
    "time"
)

type Task struct {
    value      int
    executedBy string
}

var total int
var task Task
var lock sync.Mutex

func main() {
    fmt.Printf("synchronizing goroutines demo\n")
    total = 0
    task.value = 0
    task.executedBy = ""
    display()
    // run background
    go calculate()
    go perform()
    // press ENTER to exit
    var input string
    fmt.Scanln(&input)
    fmt.Println("done")
}

```



```go
func calculate() {
    for total < 10 {
        lock.Lock()
        task.value = rand.Intn(100)
        task.executedBy = "from calculate()"
        display()
        total++
        lock.Unlock()
        time.Sleep(500 * time.Millisecond)
    }
}

func perform() {
    for total < 10 {
        lock.Lock()
        task.value = rand.Intn(100)
        task.executedBy = "from perform()"
        display()
        total++
        lock.Unlock()
        time.Sleep(500 * time.Millisecond)
    }
}

func display() {
    fmt.Println("-------------------------")
    fmt.Println(task.value)
    fmt.Println(task.executedBy)
    fmt.Println("-------------------------")
}
```

Prof. Alejandro Bello Ruiz

Pando, 08 de abril de 2022.