

```
result := make([]int, (leftLength + rightLength))  
li := 0  
ri := 0  
resulti := 0
```

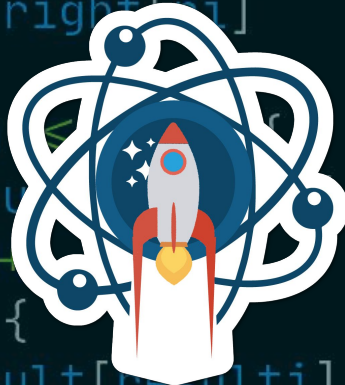
A continuación: Introducción Go y al Cómputo Concurrente

By David Bolaños

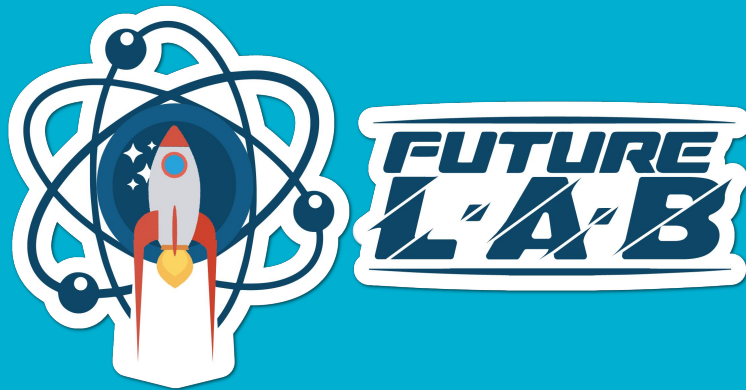
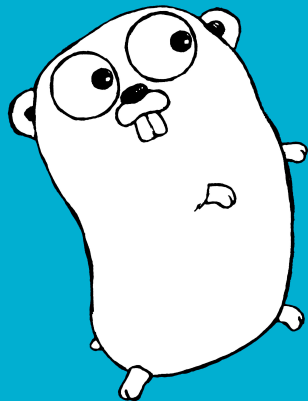
```
var rnum, lnum int
```

```
for li < leftLength || ri < rightLength {  
    if li < leftLength && ri < rightLength {  
        lnum = left[li]  
        rnum = right[ri]
```

```
        if lnum < rnum {  
            result[resulti] = lnum  
            li++  
        } else {  
            result[resulti] = rnum  
            ri++
```



Introducción a Go y al Cómputo Concurrente



Future Lab: Orbit Day

¡Hola! Soy David

Twitter: @jdbr99

Email: juan.bolanos@cimat.mx



Background Histórico

Creación

Go fue diseñado en 2007 en Google por Robert Griesemer, Rob Pike, y Ken Thompson.

Principales objetivos:

- Concurrency
- Legibilidad y Usabilidad
- Tipado estático y chequeo de errores

¿Por qué Go?

¿Por qué Go?

- Sencillo (Sólo 25 palabras reservadas)

¿Por qué Go?

- Sencillo (Sólo 25 palabras reservadas)
- Manejo de Memoria Automático

¿Por qué Go?

- Sencillo (Sólo 25 palabras reservadas)
- Manejo de Memoria Automático
- Eficiente

¿Por qué Go?

- Sencillo (Sólo 25 palabras reservadas)
- Manejo de Memoria Automático
- Eficiente
- Legibilidad

¿Por qué Go?

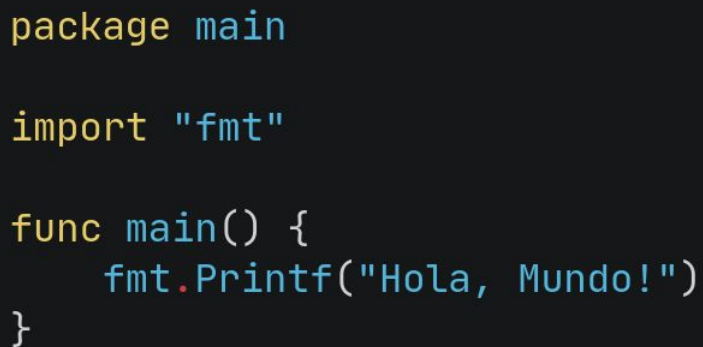
- Sencillo (Sólo 25 palabras reservadas)
- Manejo de Memoria Automático
- Eficiente
- Legibilidad
- Incita el uso de buenas prácticas

¿Por qué Go?

- Sencillo (Sólo 25 palabras reservadas)
- Manejo de Memoria Automático
- Eficiente
- Legibilidad
- Incita el uso de buenas prácticas
- Una stl **MUY** rica

Repaso

Boilerplate



```
package main

import "fmt"

func main() {
    fmt.Printf("Hola, Mundo!")
}
```

Declaración de Variables



```
miNumero := 9
```

```
var nombre string  
nombre = "David"
```

```
var goEsChido bool = true
```

Funciones I



```
func saluda(name string) {  
    fmt.Println("Hola, ", name)  
}
```


Funciones II



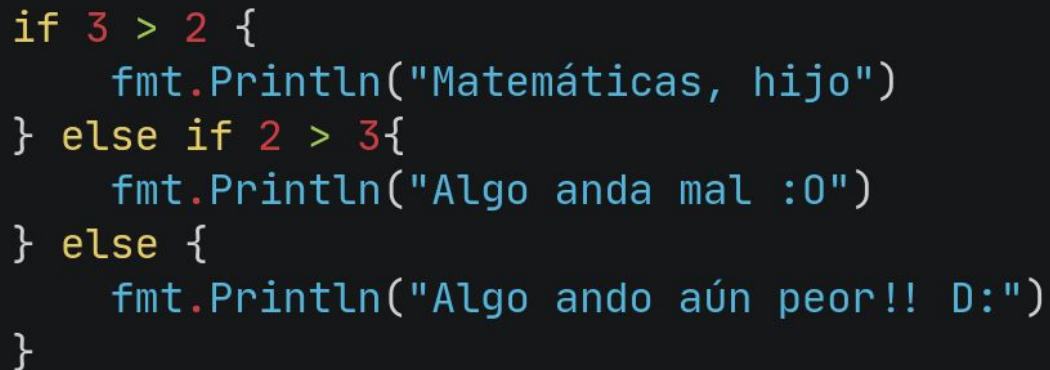
```
func sumaDos(x, y int) int {  
    return x + y  
}
```

Funciones III



```
func dobleXY(x, y float32) (float32, float32) {  
    return x*2, y*2  
}
```

Condicionales



```
if 3 > 2 {  
    fmt.Println("Matemáticas, hijo")  
} else if 2 > 3 {  
    fmt.Println("Algo anda mal :0")  
} else {  
    fmt.Println("Algo ando aún peor!! D:")  
}
```

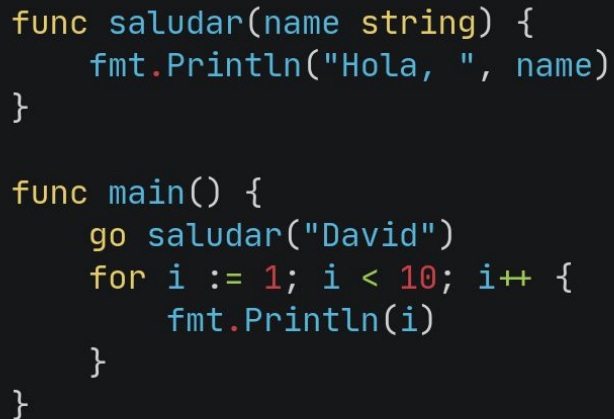
Ciclo(s)



```
fmt.Println("Ciclo `for` clásico")
for i := 0; i < 10; i++ {
    fmt.Println(i)
}

fmt.Println("Utilizando `for` como `while`")
i := 0
for i < 9 {
    fmt.Println(i * i)
    i++
}
```

GOroutines



```
func saludar(name string) {  
    fmt.Println("Hola, ", name)  
}  
  
func main() {  
    go saludar("David")  
    for i := 1; i < 10; i++ {  
        fmt.Println(i)  
    }  
}
```

make()



```
make([]int, 10)  
make([]float, 0, 10)  
make(chan int)
```

Canales




```
numberChannel := make(chan int)
```

```
numberChannel ← 9
```

```
received := ←numberChannel
```

```
fmt.Println(received)
```

Canales

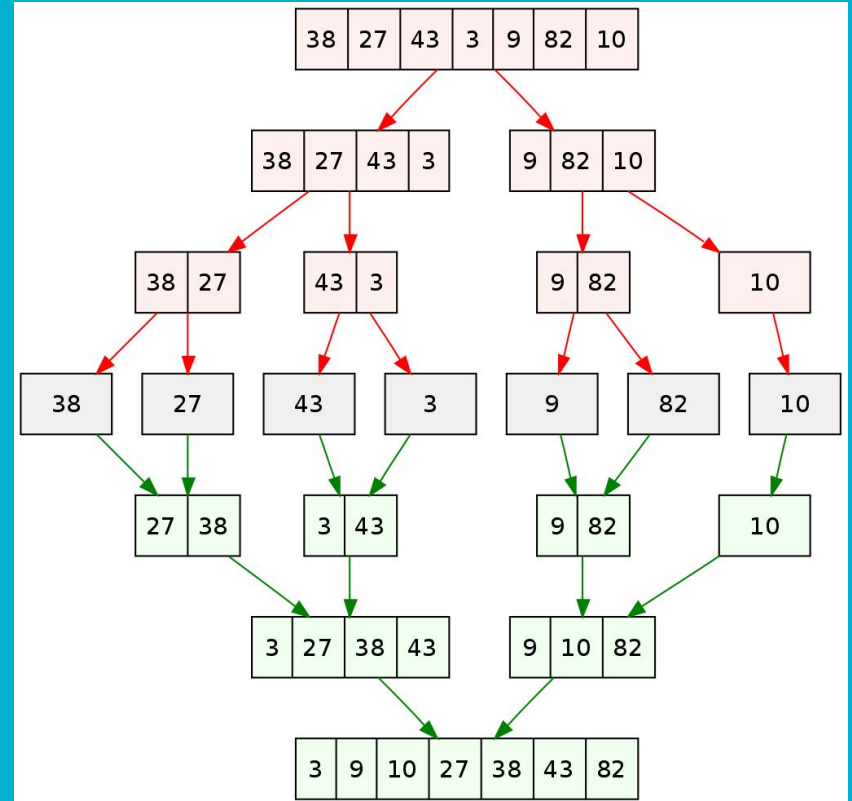


```
func sendNumber(n int, numChan chan int) {  
    numChan ← n  
}  
  
func main() {  
    numberChannel := make(chan int)  
  
    go sendNumber(9)  
  
    received := ←numberChannel  
  
    fmt.Println(received)  
}
```


Ejemplo Práctico

Mergesort

1. Dividir el arreglo en dos
2. Aplicar Mergesort a la primera mitad
3. Aplicar Mergesort a la segunda mitad
4. Juntar en orden las dos mitades



¡Gracias por su atención!

Atribuciones

La mascota “Go Gopher” fue diseñada por Renee French (<http://reneefrench.blogspot.com/>). Dicho diseño tiene licencia Creative Commons 3.0 Attributions Licence. Para más información, lee este artículo: <https://blog.golang.org/gopher>.