1.
¿Cuál es la empresa detrás del lenguage Go?
Google
2.
Para imprimir textos especiales como el Chino, requieres importar paquetes adicionales al paquete fmt
Falso
3.
¿Para qué sirven las goroutines?
Para implementar concurrencia en Go.
4.
¿La función main se ejecuta en una goroutine?
Verdadero
5.
¿Qué keyword usamos para ejecutar una función de manera concurrente?
go
6.
¿Qué imprime esta línea?
`Go package main import "fmt" func main(){ fmt.println("Hola") }`
undefined: fmt.println
7.
¿Con qué manejador de paquetes instalamos librerías de terceros?
go get
8.
¿Qué imprime esta línea `fmt.Printf("2, %T", 2)`?
2, int
9.
¿Cómo se debe llamar la función para personalizar el output en consola en los structs?
String()
10.
¿Cómo declarar correctamente arrays y slices de números enteros?
`[2]int{1,2}` es un array y `[]int{1,2}` es un slice
11.
¿Con cuál keyword podemos manejar múltiples channels?
select
12.
¿En qué momento usar switch con condición y sin condición?
Con condición cuando evaluamos el valor de una variable. Sin condición cuando queremos usar operadores lógicos.
13.
Partiendo del siguiente código, ¿qué son b y c?
`a := 10 b := &a c := *b`
`a`, `c` son números enteros y `b` es un puntero.
14.
¿Cómo declararías un map en Go que según el año guarde el nombre de los graduados del Curso?
make(map[int][]string)
15.
¿Así como en muchos otros lenguajes, para ciclos iterativos, en Go tienes `for`, `do while` y `while`?
Falso
16.
¿Cuál es la forma correcta de editar código de librerías externas?
Hacer `git clone` del código y usar `go mod replace`.
17.
¿Por qué usar `go modules`?
Porque facilita la exportación y manejo de nuestro código.
18.

¿Cuál es la principal razón por la que debemos crear nuestro `GOPATH`?
Porque es el entorno donde desarrollamos código y se descargan paquetes adicionales.
19.
¿Qué sucede si declaramos una variables que no usamos?
El código no se ejecutará.
20.
¿Qué imprime una variable cuando la inicializamos pero no le asignamos ningún valor?
Imprime su respectivo `zero value`.
21.
Con los channels podemos manejar datos entre goroutines
Verdadero
22.
Con listas de interfaces podemos guardar diferentes tipos de datos
Verdadero
23.
Los structs son estructuras de datos a los cuales no se les pueden aplicar métodos
Falso
24.
¿Qué sucede si buscamos un key que no existe en un map?
Retornará su respectivo `zero value`.
25.
¿Qué hace esta función?
```Go
func doubleReturn(a int) (c, d int) { return a, a * 2 }
```
Recibe un entero. Retorna dos enteros: El valor de entrada y el doble del valor de entrada.
26.
Al momento de declarar funciones, variables y structs no es importante si la primera letra es mayúscula o minúscula
Falso
27.
¿Es obligatorio implementar camelCase al momento de declarar variables y constantes en Go?
No es obligatorio, pero es una buena práctica.
28.
¿Qué hacemos si una función retorna 2 valores, pero solo nos interesa uno?
Guardamos el valor que nos interesa y el otro valor lo escapamos con un `_`.
29.
¿Cómo capturamos los errores si estos ocurren al usar una función?
Con estas líneas `Go if err != nil { log.Fatal(err) }`
30.
Para un valor que sabemos siempre será positivo y estará en un rango de 130 a 200, ¿cuál sería el tipo de dato más óptimo?
uint8