

GOLANG EXERCISE

1. Cara install golang

- Buka web Golang : <https://go.dev/dl/>
- Install Golang
- Cek version Golang dengan go version di terminal.

2. Cara membuat program hello world

- Create New Folder Belajar Golang
- Create File index.go
- Place Code :

```
package main
import "fmt"
func main() {
    fmt.Println("Hello World !")
}
```

- Compile melalui terminal dengan :
 - Go run index.go, atau
 - Go build index.go

3. Mengenal tipe data number

- Go memiliki 2 tipe data number yaitu int dan uint. Berikut pembagiannya

Tipe Data	Nilai Minimum	Nilai Maksimum
Int8	-128	127
Int16	-32768	32767
Int32	-2147483648	214748...
Int64	-92233...	92233720...

- Place Code

```
package main

import "fmt"

func main() {
    fmt.Println("Satu = ", 1)
    fmt.Println("Dua = ", 2)
    fmt.Println("Tiga Koma Lima = ", 3.5)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Satu = 1
Dua = 2
Tiga Koma Lima = 3.5
```

4. Mengenal tipe data Boolean

- Place Code

```
package main

import "fmt"

func main() {
    fmt.Println("Benar = ", true)
    fmt.Println("Salah = ", false)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Benar = true
Salah = false
```

5. Mengenal tipe data String

- Function untuk String

Function	Keterangan
Len("string")	Menghitung jumlah karakter pada string.
"string"[number]	Mengambil karakter pada posisi yang ditentukan.

- Place Code

```
package main

import "fmt"

func main() {
    fmt.Println("Raka Putra Eshardiansyah")
    fmt.Println(len("Raka Putra Eshardiansyah"))
    fmt.Println("Raka Putra Eshardiansyah"[0])
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Raka Putra Eshardiansyah
24
82
```

6. Variabel dalam Golang

- Variabel golang perlu untuk didefinisikan terlebih dahulu sebelum digunakan.
- Ketika inisialisasi variable wajib untuk mendefinisikan tipe datanya.
- Place Code

```
func main() {  
    var name string  
  
    name = "Raka Putra Eshardiansyah"  
    fmt.Println(name)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
Raka Putra Eshardiansyah
```

- Atau bisa melakukan Place Code seperti

```
package main  
  
import "fmt"  
  
func main() {  
    var name = "Raka Putra Eshardiansyah"  
    fmt.Println(name)  
}
```

- Atau bisa melakukan Place Code seperti

```
package main  
  
import "fmt"  
  
func main() {  
    name := "Raka Putra Eshardiansyah"  
    fmt.Println(name)  
  
    name = "Programmer Golang"  
    fmt.Println(name)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
Raka Putra Eshardiansyah  
Programmer Golang
```

- Golang juga dapat mendeklarasikan beberapa variable sekaligus.
- Place Code

```
package main

import "fmt"

func main() {
    var(
        name = "Raka Putra Eshardiansyah"
        exercise = "Belajar Golang"
    )

    fmt.Println(name)
    fmt.Println(exercise)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Raka Putra Eshardiansyah
Programmer Golang
```

7. Constant dalam Golang

- Variabel konstan tidak dapat di replace setelah di deklarasikan
- Place Code

```
package main

import "fmt"

func main() {
    const lastName = "Eshardiansyah"
    const age = 23

    //Konstan = Data tidak dapat diubah jika telah didefinisikan
    fmt.Println("Nama Akhir : ", lastName)
    fmt.Println("Umur : ", age)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Nama Akhir : Eshardiansyah
Umur : 23
```

8. Konversi Tipe Data

- Konversi dapat dilakukan salah satunya untuk mengubah variable int.
- Place Code

```
package main

import "fmt"

func main() {
    var nilai32 int32 = 100000
    var nilai64 int64 = int64(nilai32)
    var nilai8 int8 = int8(nilai64)

    fmt.Println(nilai32)
    fmt.Println(nilai64)
    fmt.Println(nilai8)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
100000
100000
-96
```

- Contoh lainnya untuk Place Code

```
package main

import "fmt"

func main() {
    var name = "Raka Putra Eshardiansyah"
    var e byte = name[0]
    var conversion string = string(e)

    fmt.Println(name)
    fmt.Println(e)
    fmt.Println(conversion)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Raka Putra Eshardiansyah
82
R
```

9. Type Declarations

- Type Declarations digunakan untuk membuat alias untuk suatu type data
- Place Code

```
package main

import "fmt"

func main() {
    type noKTP string
    var noKTPRaka noKTP = "1234567890"
    fmt.Println(noKTPRaka)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run type-declarations.go
1234567890
```

10. Operasi Matematika

- Untuk operasi matematika hampir sama dengan Bahasa pemrograman lain
- Place Code

```
package main

import "fmt"

func main() {
    var a = 10
    var b = 10
    var c = a * b
    fmt.Println(c)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
100
```

11. Operasi Perbandingan

- Operasi perbandingan juga sama dengan Bahasa pemrograman lain

- Place Code

```
package main

import "fmt"

func main() {
    var name1 = "Raka"
    var name2 = "Raka"
    var result = name1 == name2

    fmt.Println(result)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
true
```

12. Operasi Boolean

- Digunakan untuk pengkodisian
- Place Code

```
package main

import "fmt"

func main() {
    var nilaiAkhir = 80
    var absensi = 100

    var nilaiKKM = 75

    if (nilaiAkhir+absensi)/2 > nilaiKKM {
        fmt.Println("Lulus")
    }
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Lulus
```

13. Tipe data Array

- Sekumpulan data yang ditampung didalam total deklarasi
- Place Code

```
package main

import "fmt"

func main() {
    var names [3]string
    names[0] = "Raka"
    names[1] = "Putra"
    names[2] = "Eshardiansyah"

    fmt.Println(names[0])
    fmt.Println(names[1])
    fmt.Println(names[2])
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Raka
Putra
Eshardiansyah
```

14. Tipe data Slice

- Tipe data slice adalah potongan dari array
- Ukuran slice bisa berubah
- Slice dan Array selalu terkoneksi
- Membuat Slice dari Array

Membuat Slice	Keterangan
Array[low:high]	Membuat slice dimulai dari indeks low – indeks high
Array[low:]	Membuat slice dimulai dari indeks low - ???
Array[:high]	Membuat slice dimulai dari indeks 0 – indeks high
Array[:]	Membuat slice dimulai dari indeks 0 - ???

- Perbedaan penulisan Array dan Slice

```
iniArray := [5]int{1, 2, 3, 4, 5}
iniSlice := []int{1, 2, 3, 4, 5}
```


- Place Code

```
var months = [...]string{
    "Januari",
    "Februari",
    "Maret",
    "April",
    "Mei",
    "Juni",
    "Juli",
    "Agustus",
    "September",
    "Oktober",
    "November",
    "Desember"}

var slice2 = months[10:]
fmt.Println(slice2)

var slice3 = append(slice2, "Syawal")
slice3[0] = "Ramadhan"
fmt.Println(slice3)
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
[November Desember]
[Ramadhan Desember Syawal]
```

15. Tipe data Map

- Map digunakan untuk membuat key dan values pada 1 validasi
- Function manipulation yang dapat digunakan di Map

Function Map

Operasi	Keterangan
len(map)	Untuk mendapatkan jumlah data di map
map[key]	Mengambil data di map dengan key
map[key] = value	Mengubah data di map dengan key
make(map[TypeKey]TypeValue)	Membuat map baru
delete(map, key)	Menghapus data di map dengan key

- Place Code

```
book := map[string]string{
    "title": "Novel Terbaru 2023",
    "author": "Lily",
    "sinopsis": "..."}

fmt.Println(len(book))
fmt.Println(book["sinopsis"])
fmt.Println("")

//Template delete(book, [nama_key])
delete(book, "sinopsis")

fmt.Println(len(book))
fmt.Println(book["sinopsis"])
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
3
...
2
```

16. Penggunaan IF Expression

- IF digunakan untuk percabangan di Golang.
- Place Code

```
name := "Eshardiansyah"

if name == "raka" {
    fmt.Println("Hai Raka, Welcome !")
} else {
    fmt.Println("Are u a traitor ?")
}

//If short statement
var length = len(name)
if length > 5 {
    fmt.Println("Terlalu panjanggg !")
} else {
    fmt.Println("Nama sudah benar !")
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Hai Raka, Welcome !
Nama sudah benar !
PS D:\Belajar\Belajar Golang> go run index.go
Are u a traitor ?
Terlalu panjanggg !
```

17.Switch Expression

- Selain menggunakan if untuk percabangan dapat menggunakan switch expression.
- Place Code

```
name := "Admin"

switch name {
case "Raka":
    fmt.Println("Halo Raka, Welcome !")
case "Admin":
    fmt.Println("Admin is a Second Hand of Raka !")
default:
    fmt.Println("Who are you ? Get out of here.")
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Who are you ? Get out of here.
PS D:\Belajar\Belajar Golang> go run index.go
Halo Raka, Welcome !
PS D:\Belajar\Belajar Golang> go run index.go
Admin is a Second Hand of Raka !
```

- Switch juga dapat digunakan untuk percabangan Boolean
- Place Code

```
name := "Adminnnnn"
var lengthName = len(name)

switch lengthName > 5 {
case true:
    fmt.Println("Nama terlalu panjang")
case false:
    fmt.Println("Nama sudah cocok")
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Nama terlalu panjang
```

- Another Place Code

```
name := "Kara"
lengthName := len(name)

switch {
case lengthName > 10:
    fmt.Println("Nama panjang sekali, harus ganti nama")
case lengthName > 5:
    fmt.Println("Nama terlalu panjang")
default:
    fmt.Println("Nama sudah cocok !")
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Nama panjang sekali, harus ganti nama
PS D:\Belajar\Belajar Golang> go run index.go
Nama terlalu panjang
PS D:\Belajar\Belajar Golang> go run index.go
Nama sudah cocok !
```

18. For Loops di Golang

- For loops digunakan untuk mengulang
- Place Code

```
for counter := 0; counter < 10; counter++ {
    fmt.Println("Perulangan Ke-", counter)
}
```

- Output

```
Perulangan Ke- 0
Perulangan Ke- 1
Perulangan Ke- 2
Perulangan Ke- 3
Perulangan Ke- 4
Perulangan Ke- 5
Perulangan Ke- 6
Perulangan Ke- 7
Perulangan Ke- 8
Perulangan Ke- 9
```

- For loops juga bisa digunakan untuk membaca nilai di dalam Slice.
- Place Code

```
sliceName := []string{"Raka", "Putra", "Eshardiansyah"}

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

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Raka
Putra
Eshardiansyah
```

- For Loops juga dapat digunakan untuk mendapatkan nilai di MAP
- Place Code

```
book := map[string]string{
    "title": "Komik Naruto Vol.270",
    "author": "Masashi Kishimoto",
    "sinopsis": "Naruto"}

for key, value := range book {
    fmt.Println(key, "=", value)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
sinopsis = Naruto
title = Komik Naruto Vol.270
author = Masashi Kishimoto
```

19. Break & Continue di Golang

- Penjelasan

Break & Continue

- Break & continue adalah kata kunci yang bisa digunakan dalam perulangan
- Break digunakan untuk menghentikan seluruh perulangan
- Continue adalah digunakan untuk menghentikan perulangan yang berjalan, dan langsung melanjutkan ke perulangan selanjutnya

- Place Code

```
for i := 0; i < 10; i++ {
    fmt.Println("Perulangan ke-", i)
    if i == 5 {
        break
    }
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Perulangan ke- 0
Perulangan ke- 1
Perulangan ke- 2
Perulangan ke- 3
Perulangan ke- 4
Perulangan ke- 5
```

- Place Code

```
for i := 0; i < 10; i++ {
    if i%2 == 0 {
        continue
    }
    fmt.Println("Perulangan ke-", i)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Perulangan ke- 1
Perulangan ke- 3
Perulangan ke- 5
Perulangan ke- 7
Perulangan ke- 9
```

20.Function di Golang

- Penjelasan

Function

- Sebelumnya kita sudah mengenal sebuah function yang wajib dibuat agar program kita bisa berjalan, yaitu function main
- Function adalah sebuah blok kode yang sengaja dibuat dalam program agar bisa digunakan berulang-ulang
- Cara membuat function sangat sederhana, hanya dengan menggunakan kata kunci func lalu diikuti dengan nama function nya dan blok kode isi function nya
- Setelah membuat function, kita bisa mengeksekusi function tersebut dengan memanggilnya menggunakan kata kunci nama function nya diikuti tanda kurung buka, kurung tutup

- Place Code

```
func sayMyName() {
    fmt.Println("Halo saya Raka")
}

func main() {
    sayMyName()
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Halo saya Raka
PS D:\Belajar\Belajar Golang> go run index.go
Halo saya Raka
```

21.Function dengan Parameter di Golang

- Penjelasan

Function Parameter

- Saat membuat function, kadang-kadang kita membutuhkan data dari luar, atau kita sebut parameter.
- Kita bisa menambahkan parameter di function, bisa lebih dari satu
- Parameter tidaklah wajib, jadi kita bisa membuat function tanpa parameter seperti sebelumnya yang sudah kita buat
- Namun jika kita menambahkan parameter di function, maka ketika memanggil function tersebut, kita wajib memasukkan data ke parameternya

- Place Code

```
func sayMyName(name string, age int) {  
    fmt.Println("Halo saya", name, ". Umur saya", age)  
}  
  
func main() {  
    sayMyName("Raka Putra Eshardiansyah", 23)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
Halo saya Raka Putra Eshardiansyah . Umur saya 23
```

22.Function Return Value di Golang

- Penjelasan

Function Return Value

- Function bisa mengembalikan data
- Untuk memberitahu bahwa function mengembalikan data, kita harus menuliskan tipe data kembalian dari function tersebut
- Jika function tersebut kita deklarasikan dengan tipe data pengembalian, maka wajib di dalam function nya kita harus mengembalikan data
- Untuk mengembalikan data dari function, kita bisa menggunakan kata kunci return, diikuti dengan datanya

- Place Code

```
func sayMyName(firstName string, lastName string) string {  
    return "Halo saya awal saya " + firstName + " dan nama akhir saya " + lastName  
}  
  
func main() {  
    result := sayMyName("Raka", "Eshardiansyah")  
    fmt.Println(result)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Halo saya awal saya Raka dan nama akhir saya Eshardiansyah
```

23.Returning Multiple Values di Golang

- Penjelasan
- Place Code

```
func sayMyName(name string, age int) (string, int) {
    return name, age
}

func main() {
    fullName, presentAge := sayMyName("Raka Putra Eshardiansyah", 23)
    fmt.Println(fullName)
    fmt.Println(presentAge)

    //Menghiraukan return value
    fullName, _ = sayMyName("Karaaaes", 23)
    fmt.Println(fullName)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Raka Putra Eshardiansyah
23
Karaaaes
```

24.Named Return Values di Golang

- Penjelasan

Named Return Values

- Biasanya saat kita memberi tahu bahwa sebuah function mengembalikan value, maka kita hanya mendeklarasikan tipe data return value di function
- Namun kita juga bisa membuat variable secara langsung di tipe data return function nya

- Place Code

```
func getFullName() (firstName string, middleName string, lastName string, age
int) {
    firstName = "Raka"
    middleName = "Putra"
    lastName = "Eshardiansyah"
    age = 23
    return
}

func main() {
    firstName, middleName, lastName, _ := getFullName()
    fmt.Println(firstName)
    fmt.Println(middleName)
    fmt.Println(lastName)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Raka
Putra
Eshardiansyah
```

25. Variadic Function di Golang

- Penjelasan

Variadic Function

- Parameter yang berada di posisi terakhir, memiliki kemampuan dijadikan sebuah varargs
- Varargs artinya datanya bisa menerima lebih dari satu input, atau anggap saja semacam Array.
- Apa bedanya dengan parameter biasa dengan tipe data Array?
 - Jika parameter tipe Array, kita wajib membuat array terlebih dahulu sebelum mengirimkan ke function
 - Jika parameter menggunakan varargs, kita bisa langsung mengirim data nya, jika lebih dari satu, cukup gunakan tanda koma

- Place Code

```
func sumAll(numbers ...int) int {
    total := 0
    for _, value := range numbers {
        total += value
    }
    return total
}

func main() {
    total := sumAll(10, 10, 10)
    fmt.Println(total)

    slice := []int{10, 25, 20}
    total = sumAll(slice...)
    fmt.Println(total)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
30
55
```

26.Function Value di Golang

- Penjelasan



Function Value

- Function adalah first class citizen
- Function juga merupakan tipe data, dan bisa disimpan di dalam variable

- Place Code

```
func getGoodbye(name string) string {
    return "Good Bye " + name
}

func main() {
    sayGoodbye := getGoodbye
    result := sayGoodbye("Raka")
    fmt.Println(result)
    //Or call like this
    fmt.Println(getGoodbye("Raka"))
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Good Bye Raka
Good Bye Raka
```

27.Function as Parameter di Golang

- Penjelasan

Function as Parameter

- Function tidak hanya bisa kita simpan di dalam variable sebagai value
- Namun juga bisa kita gunakan sebagai parameter untuk function lain

- Place Code

```
type Filter func(string) string

func sayHelloWithFilter(name string, filter Filter) {
    nameFiltered := filter(name)
    fmt.Println("Hello", nameFiltered)
}

func spamFilter(name string) string {
    if name == "Anjing" {
        return "..."
    } else {
        return name
    }
}

//Cara penggunaan
func main() {
    sayHelloWithFilter("Eko", spamFilter)
    sayHelloWithFilter("Anjing", spamFilter)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Hello Eko
Hello ...
```

28. Anonymous Function di Golang

- Penjelasan

Anonymous Function

- Sebelumnya setiap membuat function, kita akan selalu memberikan sebuah nama pada function tersebut
- Namun kadang ada kalanya lebih mudah membuat function secara langsung di variable atau parameter tanpa harus membuat function terlebih dahulu
- Hal tersebut dinamakan anonymous function, atau function tanpa nama

- Place Code

```
type Blacklist func(string) bool

func registerUser(name string, blacklist Blacklist) {
    if blacklist(name) {
        fmt.Println("You are blocked", name)
    } else {
        fmt.Println("Welcome", name)
    }
}

func main() {
    //Anonymous function, function tanpa nama
    blacklist := func(name string) bool {
        return name == "admin"
    }
    registerUser("Raka", blacklist)
    registerUser("admin", blacklist)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Welcome Raka
You are blocked admin
```

29. Recursive Function di Golang

- Penjelasan

Recursive Function

- Recursive function adalah function yang memanggil function dirinya sendiri
- Kadang dalam pekerjaan, kita sering menemui kasus dimana menggunakan recursive function lebih mudah dibandingkan tidak menggunakan recursive function
- Contoh kasus yang lebih mudah diselesaikan menggunakan recursive adalah Factorial

- Place Code

```
func factorialLoop(value int) int {
    result := 1
    for i := value; i > 0; i-- {
        result *= i //result = result * i
    }
    return result
}

func main() {
    factorial := factorialLoop(5)
    fmt.Println(factorial)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
120
```

- Another place code

```
func factorialRecursive(value int) int {
    if value == 1 {
        return 1
    } else {
        return value * factorialRecursive(value-1)
    }
}

func main() {
    factorial := factorialRecursive(5)
    fmt.Println(factorial)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
120
```

30.Closure di Golang

- Penjelasan

Closures

- Closure adalah kemampuan sebuah function berinteraksi dengan data-data disekitarnya dalam scope yang sama
- Harap gunakan fitur closure ini dengan bijak saat kita membuat aplikasi

- Place Code

```
func main() {
    counter := 0
    increment := func() {
        counter++
        fmt.Println(counter)
    }

    increment()
    increment()
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
1
2
```

31. Defer, Panic & Recover di Golang

- Penjelasan Defer

Defer

- Defer function adalah function yang bisa kita jadwalkan untuk dieksekusi setelah sebuah function selesai di eksekusi
- Defer function akan selalu dieksekusi walaupun terjadi error di function yang dieksekusi

- Place Code Defer

```
func logging() {
    fmt.Println("Selesai memanggil function")
}

func runApplication(value int) {
    defer logging()
    fmt.Println("Run Application")
    result := 10 / value
    fmt.Println("Result", result)
}

func main() {
    runApplication(5)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Run Application
Result 2
Selesai memanggil function
```

- Penjelasan Panic

Panic

- Panic function adalah function yang bisa kita gunakan untuk menghentikan program
- Panic function biasanya dipanggil ketika terjadi error pada saat program kita berjalan
- Saat panic function dipanggil, program akan terhenti, namun defer function tetap akan dieksekusi

- Place Code Panic

```
func endApp() {
    fmt.Println("Aplikasi selesai")
}

func runApp(name string) {
    defer endApp()
    if name == "Admin" {
        panic("ERROR")
    }
    fmt.Println("Aplikasi Berjalan")
}

func main() {
    runApp("Admin")
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Aplikasi selesai
panic: ERROR

goroutine 1 [running]:
main.runApp({0x808192?, 0xc00003e000?})
    D:/Belajar/Belajar Golang/index.go:12 +0xb9
main.main()
    D:/Belajar/Belajar Golang/index.go:18 +0x25
exit status 2
```

- Penjelasan Recover

Recover

- Recover adalah function yang bisa kita gunakan untuk menangkap data panic
- Dengan recover proses panic akan terhenti, sehingga program akan tetap berjalan

- Place Code Recover

```
func endApp() {
    message := recover()
    if message != nil {
        fmt.Println("Error dengan message :", message)
    }
    fmt.Println("Aplikasi selesai")
}

func runApp(name string) {
    defer endApp()
    if name == "Admin" {
        panic("ERROR")
    }
    fmt.Println("Aplikasi Berjalan")
}

func main() {
    runApp("Admin")
    fmt.Println("Raka")
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Error dengan message : ERROR
Aplikasi selesai
Raka
```

32.Struct di Golang

- Penjelasan

Struct

- Struct adalah sebuah template data yang digunakan untuk menggabungkan nol atau lebih tipe data lainnya dalam satu kesatuan
- Struct biasanya representasi data dalam program aplikasi yang kita buat
- Data di struct disimpan dalam field
- Sederhananya struct adalah kumpulan dari field

- Penjelasan data lanjutan

Membuat Data Struct

- Struct adalah template data atau prototype data
- Struct tidak bisa langsung digunakan
- Namun kita bisa membuat data/object dari struct yang telah kita buat

- Place Code

```
type Customer struct {
    name, address string
    age      int
}

func main() {
    var identitas Customer
    identitas.name = "Raka Putra Eshardiansyah"
    identitas.address = "Bekasi"
    identitas.age = 23

    fmt.Println(identitas)

    //atau panggil seperti ini
    fmt.Println(identitas.name)
    fmt.Println(identitas.address)
    fmt.Println(identitas.age)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
{Raka Putra Eshardiansyah Bekasi 23}
Raka Putra Eshardiansyah
Bekasi
23
```

- Penjelasan Struct Literals

Struct Literals

- Sebelumnya kita telah membuat data dari struct, namun sebenarnya ada banyak cara yang bisa kita gunakan untuk membuat data dari struct

- Place Code

```
joko := Customer{
    name: "Joko Pranowo",
    address: "Jakarta",
    age: 35,
}
fmt.Println(joko)
```

- Output

```
{Joko Pranowo Jakarta 35}
```

33.Struct Method

- Penjelasan

Struct Method

- Struct adalah tipe data seperti tipe data lainnya, dia bisa digunakan sebagai parameter untuk function
- Namun jika kita ingin menambahkan method ke dalam structs, sehingga seakan-akan sebuah struct memiliki function
- Method adalah function

- Place Code

```
func (customer Customer) sayHello(name string) {
    fmt.Println("Hi,", name, "My name is", customer.name)
}

func main() {
    var identitas Customer
    identitas.name = "Raka Putra Eshardiansyah"
    identitas.address = "Bekasi"
    identitas.age = 23

    identitas.sayHello("Raka")
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Hi, Raka My name is Raka Putra Eshardiansyah
```

34.Interface di Golang

- Penjelasan

Interface

- Interface adalah tipe data Abstract, dia tidak memiliki implementasi langsung
- Sebuah interface berisikan definisi-definisi method
- Biasanya interface digunakan sebagai kontrak

- Place Code

```
type HasName interface {  
    getName() string  
}  
  
func sayHello(hasName HasName) {  
    fmt.Println("Hello", hasName.getName())  
}  
  
type Person struct {  
    name string  
}  
  
func (person Person) getName() string {  
    return person.name  
}  
  
func main() {  
    var identitas Person  
    identitas.name = "Raka"  
    sayHello(identitas)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
Hello Raka
```

35.Interface Kosong di Golang

- Penjelasan

Interface Kosong

- Go-Lang bukanlah bahasa pemrograman yang berorientasi objek
- Biasanya dalam pemrograman berorientasi objek, ada satu data parent di puncak yang bisa dianggap sebagai semua implementasi data yang ada di bahasa pemrograman tersebut
- Contoh di Java ada `java.lang.Object`
- Untuk menangani kasus seperti ini, di Go-Lang kita bisa menggunakan interface kosong
- Interface kosong adalah interface yang tidak memiliki deklarasi method satupun, hal ini membuat secara otomatis semua tipe data akan menjadi implementasi nya

Penggunaan Interface Kosong di Go-Lang

- Ada banyak contoh penggunaan interface kosong di Go-Lang, seperti :
 - `fmt.Println(a ...interface{})`
 - `panic(v interface{})`
 - `recover() interface{}`
 - dan lain-lain

- Place Code

```
func Ups(i int) interface{} {  
    if i == 1 {  
        return 1  
    } else if i == 2 {  
        return true  
    } else {  
        return "ups"  
    }  
}
```



```
func main() {  
    var data interface{} = Ups(1)  
    fmt.Println(data)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
1
```

36.Nil di Golang

- Penjelasan

Nil

- Biasanya di dalam bahasa pemrograman lain, object yang belum diinisialisasi maka secara otomatis nilainya adalah null atau nil
- Berbeda dengan Go-Lang, di Go-Lang saat kita buat variable dengan tipe data tertentu, maka secara otomatis akan dibuatkan default value nya
- Namun di Go-Lang ada data nil, yaitu data kosong
- Nil sendiri hanya bisa digunakan di beberapa tipe data, seperti interface, function, map, slice, pointer dan channel

- Place Code

```
func newMap(name string) map[string]string {
    if name == "" {
        return nil
    } else {
        return map[string]string{
            "name": name
        }
    }
}

func main() {
    var person map[string]string = newMap("")

    if person == nil {
        fmt.Println("Data Kosong")
    } else {
        fmt.Println(person)
    }
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Data Kosong
```

37.error Interface di Golang

- Penjelasan

Membuat Error

- Untuk membuat error, kita tidak perlu manual.
- Go-Lang sudah menyediakan library untuk membuat helper secara mudah, yang terdapat di package errors (Package akan kita bahas secara detail di materi tersendiri)

- Place Code

```
import (  
    "errors"  
    "fmt"  
)  
  
func Pembagi(nilai int, pembagi int) (int, error) {  
    if pembagi == 0 {  
        return 0, errors.New("Pembagi tidak boleh 0")  
    } else {  
        result := nilai / pembagi  
        return result, nil  
    }  
}  
  
func main() {  
    hasil, err := Pembagi(100, 0)  
    if err == nil {  
        fmt.Println("Hasil", hasil)  
    } else {  
        fmt.Println("Error:", err.Error())  
    }  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
Error: Pembagi tidak boleh 0
```

38.Type Assertions di Golang

- Penjelasan

Type Assertions

- Type Assertions merupakan kemampuan merubah tipe data menjadi tipe data yang diinginkan
- Fitur ini sering sekali digunakan ketika kita bertemu dengan data interface kosong

- Place Code

```
func random() interface{} {  
    return "This is String"  
}  
  
func main() {  
    result := random()  
    resultFinal := result.(string)  
    fmt.Println(resultFinal)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
This is String
```

- Penjelasan Lanjutan

Type Assertions Menggunakan Switch

- Saat salah menggunakan type assertions, maka bisa berakibat terjadi panic di aplikasi kita
- Jika panic dan tidak ter-recover, maka otomatis program kita akan mati
- Agar lebih aman, sebaiknya kita menggunakan switch expression untuk melakukan type assertions

- Place Code

```
func random() interface{} {  
    return 15  
}  
  
func main() {  
    result := random()  
    switch value := result.(type) {  
    case string:  
        fmt.Println("Ini adalah String", value)  
    case int:  
        fmt.Println("Ini adalah Int", value)  
    default:  
        fmt.Println("Tidak termasuk type data apapun")  
    }  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
Ini adalah Int 15
```

39.Pointer di Golang

- Penjelasan

Pass by Value

- Secara default di Go-Lang semua variable itu di passing by value, bukan by reference
- Artinya, jika kita mengirim sebuah variable ke dalam function, method atau variable lain, sebenarnya yang dikirim adalah duplikasi value nya

- Place Code

```
type Address struct {
    city, province, country string
}

func main() {
    address1 := Address{"Jakarta", "Bekasi", "Bandung"}
    address2 := address1
    address2.city = "Karawang"

    fmt.Println(address1)
    fmt.Println(address2)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
{Jakarta Bekasi Bandung}
{Karawang Bekasi Bandung}
```

- Penjelasan Pointer

Pointer

- Pointer adalah kemampuan membuat reference ke lokasi data di memory yang sama, tanpa menduplikasi data yang sudah ada
- Sederhananya, dengan kemampuan pointer, kita bisa membuat pass by reference

Operator &

- Untuk membuat sebuah variable dengan nilai pointer ke variable yang lain, kita bisa menggunakan operator & diikuti dengan nama variable nya

- Place Code

```
type Address struct {
    city, province, country string
}

func main() {
    address1 := Address{"Jakarta", "Bekasi", "Bandung"}
    address2 := &address1
    address2.city = "Karawang"

    fmt.Println(address1)
    fmt.Println(address2)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
{Karawang Bekasi Bandung}
&{Karawang Bekasi Bandung}
```

- Penjelasan Operator *

Operator *

- Saat kita mengubah variable pointer, maka yang berubah hanya variable tersebut.
- Semua variable yang mengacu ke data yang sama tidak akan berubah
- Jika kita ingin mengubah seluruh variable yang mengacu ke data tersebut, kita bisa menggunakan operator *

- Place Code

```
type Address struct {
    city, province, country string
}

func main() {
    address1 := Address{"Jakarta", "Bekasi", "Bandung"}
    address2 := &address1

    //Otomatis mengganti referer dengan pointer
    *address2 = Address{"Karawang", "Cikampek", "Purwakarta"}

    fmt.Println(address1)
    fmt.Println(address2)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
{Karawang Cikampek Purwakarta}
&{Karawang Cikampek Purwakarta}
```

- Penjelasan function new

Function new

- Sebelumnya untuk membuat pointer dengan menggunakan operator &
- Go-Lang juga memiliki function new yang bisa digunakan untuk membuat pointer
- Namun function new hanya mengembalikan pointer ke data kosong, artinya tidak ada data awal

- Place Code

```
type Address struct {
    city, province, country string
}

func main() {
    address1 := Address{"Jakarta", "Bekasi", "Bandung"}
    address2 := &address1

    //Otomatis mengganti referer dengan pointer
    *address2 = Address{"Karawang", "Cikampek", "Purwakarta"}

    //Function New
    alamat1 := new(Address)
    alamat2 := alamat1
    alamat2.city = "Purwokerto"

    fmt.Println(address1)
    fmt.Println(address2)
    fmt.Println(alamat2)
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go
{Karawang Cikampek Purwakarta}
&{Karawang Cikampek Purwakarta}
&{Purwokerto }
```

40.Pointer di Function

- Penjelasan

Pointer di Function

- Saat kita membuat parameter di function, secara default adalah pass by value, artinya data akan di copy lalu dikirim ke function tersebut
- Oleh karena itu, jika kita mengubah data di dalam function, data yang aslinya tidak akan pernah berubah.
- Hal ini membuat variable menjadi akan, karena tidak akan bisa diubah
- Namun kadang kita ingin membuat function yang bisa mengubah data asli parameter tersebut
- Untuk melakukan ini, kita juga bisa menggunakan pointer di function
- Untuk menjadikan sebuah parameter sebagai function, kita bisa menggunakan operator * di parameternya

- Place Code

```
type Address struct {  
    city, province, country string  
}  
  
func ChangeCountryToIndonesia(address *Address) {  
    address.country = "Indonesia"  
}  
  
func main() {  
    alamat := Address{"Jakarta", "DKI Jakarta", ""}  
    ChangeCountryToIndonesia(&alamat)  
    fmt.Println(alamat)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
{Jakarta DKI Jakarta Indonesia}
```

41.Pointer di Method di Golang

- Penjelasan

Pointer di Method

- Walaupun method akan menempel di struct, tapi sebenarnya data struct yang diakses di dalam method adalah pass by value
- Sangat direkomendasikan menggunakan pointer di method, sehingga tidak boros memory karena harus selalu diduplikasi ketika memanggil method

- Place Code

```
type Man struct {  
    name string  
}  
  
func (man *Man) Married() {  
    man.name = "Hello Mr." + man.name  
}  
  
func main() {  
    status := Man{"Raka"}  
    status.Married()  
    fmt.Println(status)  
}
```

- Output

```
PS D:\Belajar\Belajar Golang> go run index.go  
{Hello Mr.Raka}
```

42.GOPATH di Golang

- Penjelasan

GOPATH

- Sebelumnya saat kita membuat project aplikasi Go-Lang kita belum membahas tentang GOPATH
- GOPATH adalah environment variable yang berisikan lokasi tempat kita menyimpan project dan library Go-Lang
- GOPATH wajib di buat ketika kita mulai membuat aplikasi lebih dari satu file atau butuh menggunakan library dari luar

43.Package dan Import di Golang

- Penjelasan

Package

- Package adalah tempat yang bisa digunakan untuk mengorganisir kode program yang kita buat di Go-Lang
- Dengan menggunakan package, kita bisa merapikan kode program yang kita buat
- Package sendiri sebenarnya hanya direktori folder di sistem operasi kita