Map

Map

Sama seperti tipe data *array* dan *slice*, *map* juga berfungsi untuk menyimpan satu atau lebih data namun, *map* disimpan sebagai *key:value pairs* (pasang key dan value) .

Jika teman-teman pernah mempelajari bahasa Javascript dan PHP, maka *map* pada bahasa Go mirip seperti tipe data *object* pada Javascript dan mirip seperti tipe data *associative array* pada PHP.

Semua *key* dan *value* memiliki tipe data yang static, sehingga semua *key* harus memiliki tipe data yang sama begitu pula juga dengan tipe data *value* nya. Kemudian setiap *key* pada sebuah *map* harus unik namun *value* nya tidak mesti unik.

Map juga termasuk salah satu tipe data yang masuk dalam kategori *reference type* sama seperti tipe data *slice*. Berarti jika ada suatu *map* yang berusaha untuk meng-copy *map* lainnya, dan *map* tersebut mengganti *value* pada suatu *key*, maka value dari *map* lainnya tersebut juga akan ikut terganti.

Zero value dari tipe data *map* adalah *nil*.

Map

Ketika kita ingin membuat sebuah *map*, maka kita juga harus langsung menginisialisasikannya. Contohnya seperti pada gambar disebelah kanan.

Kita membuat sebuah variable bernama *person* yang memiliki tipe data *map*.

Penulisan map[string]string berarti tipe data *key* dari *map* harus *string* dan *value* dari *map* juga harus *string*. Lalu jika kita ingin memberikan nilai atau value kepada *map* nya, maka harus di inisialisasi terlebih dahulu.

Bisa kita lihat pada line 12, 13 dan 14, kita memberikan *value* kepada variable *person* dengan *key* yang berbeda-beda atau unik. Kita juga dapat mendapatkan *value* dari *map* dengan cara mengakses *key* dari *map* nya seperti pada line 18 - 20.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua di sebelah kanan.

```
2  package main
3
4  import "fmt"
5
6  func main() {
7
8    var person map[string]string // Deklarasi
9
10   person = map[string]string{} // Inisialisasi
11
12   person["name"] = "Airell"
13
14   person["age"] = "23"
15
16   person["address"] = "Jalan Sudirman"
17
18   fmt.Println("name:", person["name"])
19   fmt.Println("age:", person["age"])
20   fmt.Println("address:", person["address"])
21 }
```

```
name: Airell
age: 23
address: Jalan Sudirman
```

Map

Kita juga dapat memberikan *value* beserta dengan *key* nya dengan langsung. Contohnya seperti pada gambar pertama di sebelah kanan.

Jika kita perhatikan pada line 33, kita perlu menambahkan tanda koma , setelah kita menuliskan *value* nya walaupun di bawah nya sudah tidak ada *value* dan *key* baru. Hal ini dikarenakan kita menulis *map* dengan cara horizontal (ke bawah).

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

func main() {

  var person = map[string]string{
    "name":    "Airell",
    "age":     "23",
    "address": "Jalan Sudirman",
  }

  fmt.Println("name:", person["name"])
  fmt.Println("age:", person["age"])
  fmt.Println("address:", person["address"])
}
```

```
name: Airell
age: 23
address: Jalan Sudirman
```

Map (Looping with map)

Ketika kita ingin melakukan loop terhadap sebuah *map,* maka kita dapat menggunakan *range* loop. Contohnya seperti pada gambar pertama di sebelah kanan.

Kita bisa dengan mudah mendapatkan *key* dan *value* nya.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

func main() {

  var person = map[string]string{
    "name":    "Airell",
    "age":     "23",
    "address": "Jalan Sudirman",
  }

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

```
name: Airell
age: 23
address: Jalan Sudirman
```

Map (Deleting value)

Kita juga dapat menghapus *value* dari sebuah *map* dengan cara menggunakan fungsi *delete*. Contohnya seperti pada gambar di sebelah kanan. Argumen pertama yang kita berikan pada fungsi *delete* adalah *map* yang ingin kita hapus atau bisa juga variable tempat *map* disimpan. Lalu argumen kedua adalah *key* yang menyimpan *value* nya.

```go
package main

import "fmt"

func main() {

  var person = map[string]string{
    "name":    "Airell",
    "age":     "23",
    "address": "Jalan Sudirman",
  }

  fmt.Println("Before deleting:", person)

  delete(person, "age")

  fmt.Println("After deleting:", person)
}
```

Jika kita jalankan pada terminal, maka hasilnya akan seperti pada gambar kedua.

```
Before deleting: map[address:Jalan Sudirman age:23 name:Airell]
After deleting: map[address:Jalan Sudirman name:Airell]
```

Bisa kita perhatikan pada gambar kedua, setelah *key* age dihapus maka *key* dan *value* nya sudah tidak akan ada lagi pada *map* nya.

Map (Detecting a value)

Kita juga dapat mendeteksi keberadaan suatu *value* dengan cara mengakses *key* dari *map* nya lalu memberikan 2 variable sebagai penampungnya.

Seperti yang terdapat pada line 92 dan line 102 pada gambar pertama disebelah kanan. Perlu diingat disini bahwa memberikan 2 variable sebagai penampung merupakan hal opsional. Hal ini kita lakukan hanya untuk mendeteksi keberadaan suatu *value*.

Variable *value* yang kita berikan akan mengembalikan *value* asli dari *map* nya jika memang *key* nya ada, jika tidak ada maka kita akan mendapat *zero value* dari *value* nya. Kemudian variable *exist* yang kita berikan akan mengembalikan nilai dengan tipe data bool. Variable *exist* akan mengembalikan *true* jika memang *key* nya ada, jika tidak maka akan mengembalikan *false*.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

func main() {

  var person = map[string]string{
    "name":    "Airell",
    "age":     "23",
    "address": "Jalan Sudirman",
  }

  value, exist := person["age"]

  if exist {
    fmt.Println(value)
  } else {
    fmt.Println("Value does'nt exist")
  }

  delete(person, "age")

  value, exist = person["age"]

  if exist {
    fmt.Println(value)
  } else {
    fmt.Println("Value has been deleted")
  }

}
```

```
23
Value has been deleted
```

Map (Combining slice with map)

Kita juga dapat mengkombinasikan *slice* dengan *map*. Contohnya seperti pada gambar pertama di sebelah kanan. Kita juga dapat me-looping nya dengan menggunakan range loop.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

func main() {

  var people = []map[string]string{
    {"name": "Airell", "age": "23"},
    {"name": "Nanda", "age": "23"},
    {"name": "Mailo", "age": "15"},
  }

  for i, person := range people {
    fmt.Printf("Index: %d, name: %s, age: %s\n", i, person["name"], person["age"])
  }

}
```

```
Index: 0, name: Airell, age: 23
Index: 1, name: Nanda, age: 23
Index: 2, name: Mailo, age: 15
```