4 *Struct*, *Pointer*, *Slice* dan *Map* Menggunakan Bahasa Pemrograman Go

Objektif :

- **Mahasiswa dapat memahami konsep penggunaan *struct, pointer, slice* dan *map***
- **Mahasiswa dapat memahami hubungan antara *struct* dan *pointer*.**
- **Mahasiswa dapat memahami hubungan *slice* dan *map*.**

4.1 Pointer

Pointer adalah sebuah variabel yang nilainya berisi alamat memori dari variabel lain. Dalam bahasa pemrograman Golang pendeklarasian sebuah pointer ditandai dengan *** (asterik)** yang diikuti oleh tipe data dari variabel tersebut, contoh:

```
package main

import "fmt"

func main() {
  var a int
  var b *int // deklarasi  pointer
}
```

Pada Golang untuk mengambil alamat memori dari sebuah variabel dapat menggunakan tanda **& (ampersand)** yang diletakkan sebelum nama dari variabel (dalam notasi heksadesimal), contoh:

```
package main

func main() {
  var a int
  var b *int // deklarasi  pointer

  a = 5
  b = &a // b = alamat memori dari a (0xc4200xxxxx)
}
```

Sedangkan untuk mengambil atau mengubah nilai dari pointer dapat menggunakan tanda *** (asterik)** yang diletakkan sebelum nama dari variabel, contoh:

```
1  package main
 1
 2 import "fmt"
 3
 4 func main() {
 5 › var a int
 6 › var b *int // deklarasi  pointer
 7
 8 › a = 5
 9 › b = &a // b = alamat memori dari a (0xc4200xxxxx)
10 › *b = 3
11
12 › fmt.Println("nilai  a  =", a)
13 › fmt.Println("alamat a  =", &a)
14 › fmt.Println("nilai  b  =", *b)
15 › fmt.Println("alamat b  =", b)
16 }
~
~
~
~
~
~
main.go  OK 1,1
 1 ➜  4.1-deklarasi-pointer  go run main.go
 2 nilai  a  = 3
 3 alamat a  = 0×c4200140d8
 4 nilai  b  = 3
 5 alamat b  = 0×c4200140d8
```

4.2 Struct

Struct pada Golang digunakan untuk menyimpan beberapa properti atau fungsi dalam 1 wadah. Struct sering digunakan dalam pembuatan skema database, skema JSON, skema XML, dll. Untuk mendeklarasikan sebuah struct dapat menggunakan keyword **type** diikuti nama dari struct tersebut dan keyword **struct** pada Golang, contoh:

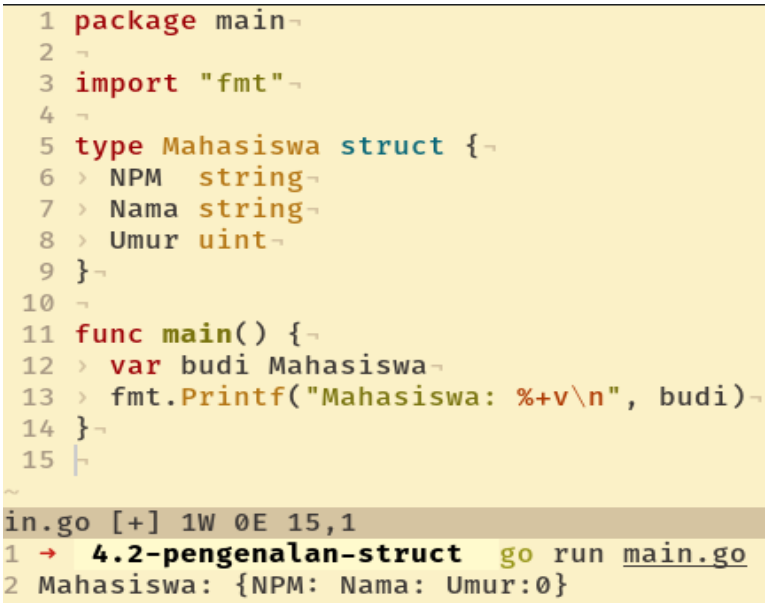
```
package main
¬
type Mahasiswa struct {
› NPM  string
› Nama string
› Umur uint
}
```

Pada gambar di atas, struct Mahasiswa memiliki 3 buah properti yaitu NPM bertipe string, Nama bertipe string, dan Umur bertipe unsigned integer (bilangan cacah, 0, 1, 2, 3, …, dst).

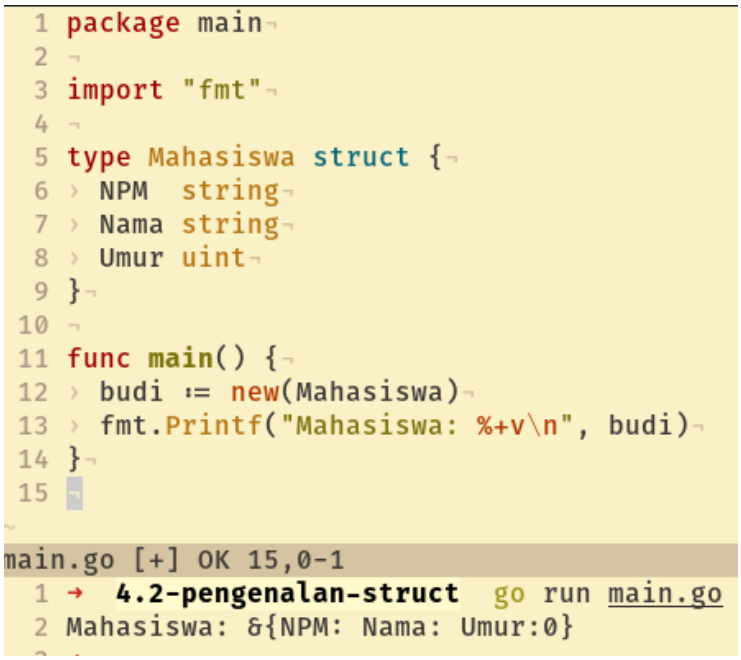
Inisialisasi struct

Inisialisasi struct pada Golang dapat dilakukan dengan 4 cara, diantaranya:

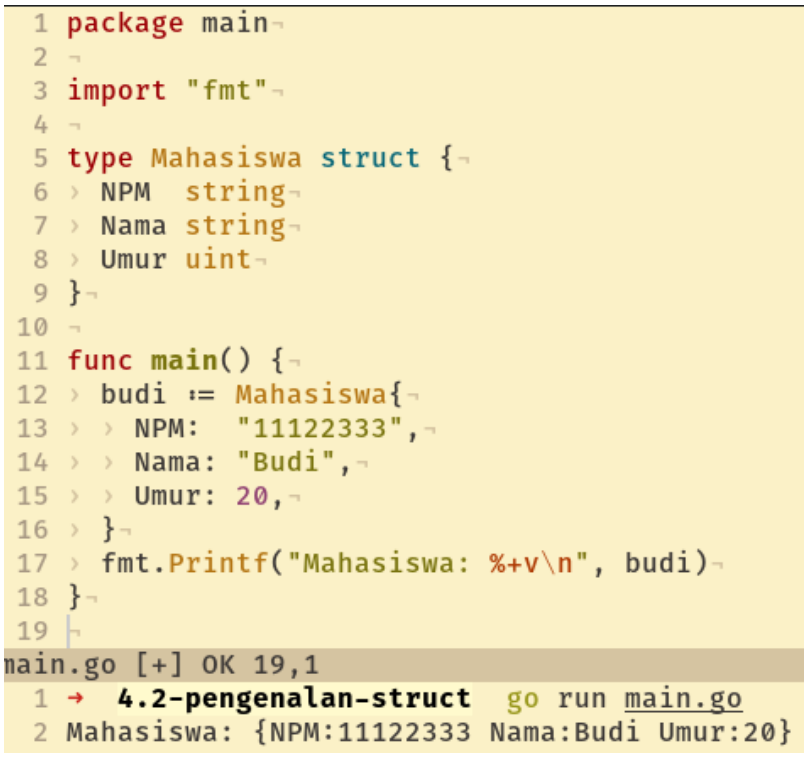
1. Menggunakan keyword var, cara akan ini membuat sebuah variabel lokal dengan nilai default pada setiap properti sama dengan kosong (0 untuk int, 0.0 untuk float, “” untuk string, nil untuk pointer, dst), contoh:



2. Menggunakan fungsi new, cara ini akan mengalokasikan memori dan mengembalikan nilai bertipe pointer dengan nilai default pada setiap propertinya sama dengan kosong (0 untuk int, 0.0 untuk float, “” untuk string, nil untuk pointer, dst), contoh:

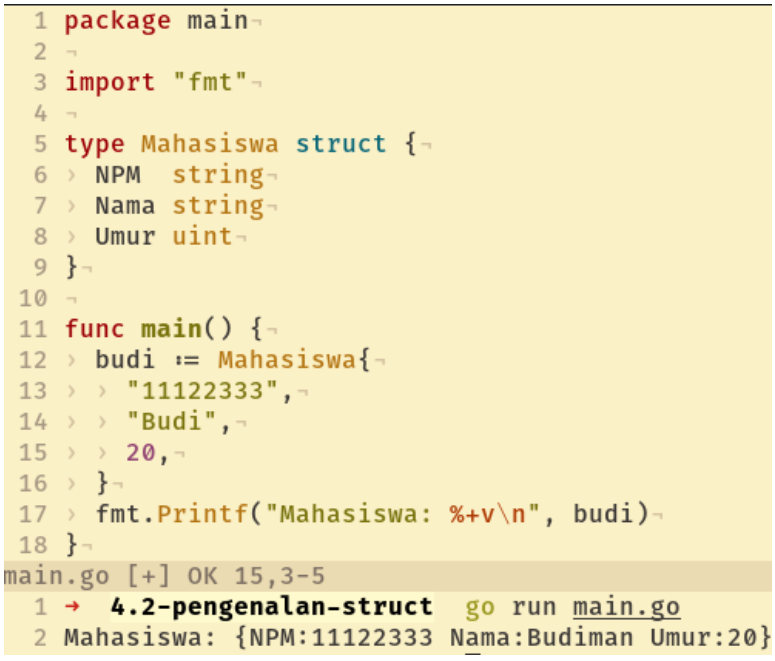


3. Memberikan nilai diikuti dengan nama dari properti tersebut, cara ini dapat dilakukan saat pendeklarasian sebuah struct. Urutan properti pada pendeklarasian struct tidak akan mempengaruhi proses pendeklarasian, contoh:



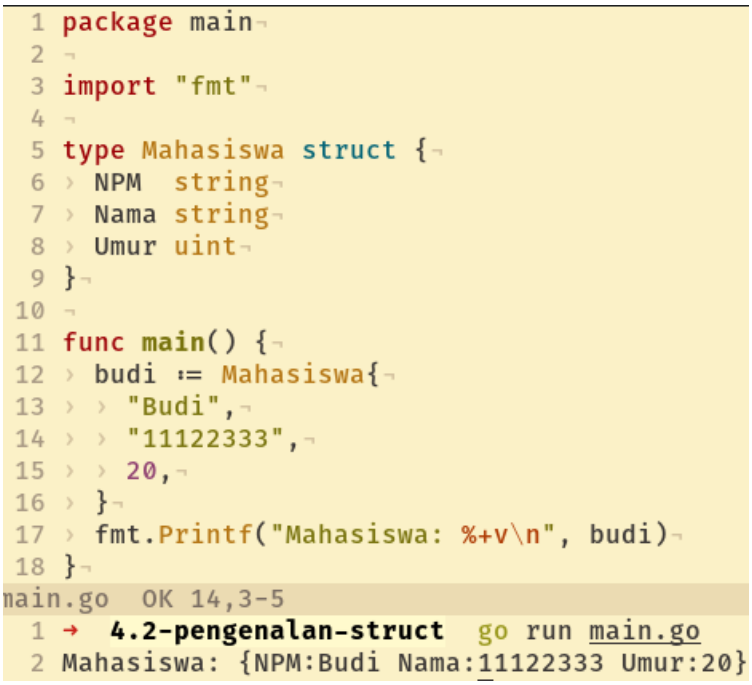
```
 1 package main
 2
 3 import "fmt"
 4
 5 type Mahasiswa struct {
 6 › NPM  string
 7 › Nama string
 8 › Umur uint
 9 }
10
11 func main() {
12 › budi := Mahasiswa{
13 › › NPM:  "11122333",
14 › › Nama: "Budi",
15 › › Umur: 20,
16 › }
17 › fmt.Printf("Mahasiswa: %+v\n", budi)
18 }
19
main.go [+] OK 19,1
 1 ➜  4.2-pengenalan-struct  go run main.go
 2 Mahasiswa: {NPM:11122333 Nama:Budi Umur:20}
```

4. Jika urutan properti sudah diketahui, nama properti dapat dihilangkan pada saat pendeklarasian struct, contoh:



```
 1 package main
 2
 3 import "fmt"
 4
 5 type Mahasiswa struct {
 6 › NPM  string
 7 › Nama string
 8 › Umur uint
 9 }
10
11 func main() {
12 › budi := Mahasiswa{
13 › › "11122333",
14 › › "Budi",
15 › › 20,
16 › }
17 › fmt.Printf("Mahasiswa: %+v\n", budi)
18 }
main.go [+] OK 15,3-5
 1 ➜  4.2-pengenalan-struct  go run main.go
 2 Mahasiswa: {NPM:11122333 Nama:Budiman Umur:20}
```

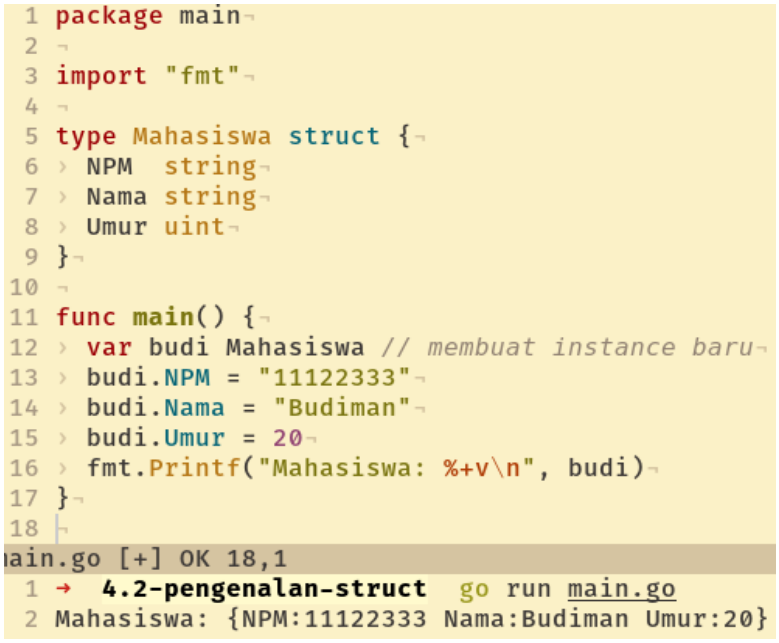
Urutan properti perlu diperhatikan pada saat menggunakan cara di atas, dikarenakan Golang akan memasukkan nilai ke dalam properti sesuai dengan urutan, berikut adalah contoh urutan yang salah:



```
 1 package main
 2
 3 import "fmt"
 4
 5 type Mahasiswa struct {
 6 › NPM  string
 7 › Nama string
 8 › Umur uint
 9 }
10
11 func main() {
12 › budi := Mahasiswa{
13 › › "Budi",
14 › › "11122333",
15 › › 20,
16 › }
17 › fmt.Printf("Mahasiswa: %+v\n", budi)
18 }
main.go  OK 14,3-5
 1 ➜  4.2-pengenalan-struct  go run main.go
 2 Mahasiswa: {NPM:Budi Nama:11122333 Umur:20}
```

Pengisian Nilai Struct

Pengisian sebuah nilai struct dapat dilakukan dengan cara mengakses properti struct terlebih dahulu. Pengaksesan properti pada struct menggunakan **. (dot)** yang diikuti oleh nama instance dan nama properti dari struct tersebut, contoh:



```
1  package main
2
3  import "fmt"
4
5  type Mahasiswa struct {
6      NPM  string
7      Nama string
8      Umur uint
9  }
10
11 func main() {
12     var budi Mahasiswa // membuat instance baru
13     budi.NPM = "11122333"
14     budi.Nama = "Budiman"
15     budi.Umur = 20
16     fmt.Printf("Mahasiswa: %+v\n", budi)
17 }
18
ain.go [+] OK 18,1
1  ➜  4.2-pengenalan-struct  go run main.go
2  Mahasiswa: {NPM:11122333 Nama:Budiman Umur:20}
```

Embedded struct

Embedded struct adalah sebuah struct yang diletakkan di dalam sebuah struct lain. Struct jenis ini merepresentasikan sebuah hubungan pada struct, contoh:

```
1  package main
2
3  import "fmt"
4
5  type Mahasiswa struct {
6      NPM             string
7      Nama            string
8      Umur            uint
9      AkunStudentSite StudentSite
10 }
11
12 type StudentSite struct {
13     Username string
14     Password string
15 }
16
17 func main() {
18     var budi Mahasiswa
19     fmt.Printf("%+v", budi)
20 }
```

Pada contoh di atas, struct Mahasiswa memiliki relasi one-to-one terhadap struct StudentSite. Karena satu mahasiswa memiliki satu akun studentsite. Untuk mendefinisikan bahwa kedua struct tersebut saling berhubungan diperlukan sebuah field yang menjembatani kedua struct tersebut, dalam hal ini field AkunStudentSite pada struct Mahasiswa.

4.3 Slice

Pengenalan *Slice*

Slice adalah *reference*/referensi elemen array, jika suatu array pada dasarnya sudah memiliki ukuran yang tetap, berbeda dengan *slice* yang ukurannya dapat secara dinamis dan mengacu secara fleksibel kepada elemen sebuah array. *Slice* dapat dibentuk, atau bisa dihasilkan dari suatu manipulasi sebuah array atau *slice* lainnya, perubahan pada data di setiap elemen *slice* dapat berdampak pada *slice* lain yang memiliki alamat memori yang sama.

Inisialisasi *Slice*

Pembuatan *slice* memilki cara yang hampir sama seperti pembuatan array, namun terdapat perbedaan saat melakukan pendefinisian jumlah elemen awal deklarasi. Berikut adalah contoh pembuatan *slice*.

```
1    var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop", "web"}
2
3    fmt.Println(kursus[0]) //dbms
```

Output dari kode diatas akan menampilkan isi dari *slice* dengan indeks pertama yaitu dbms.

Nilai Dasar Pembentukan *Slice*

Seperti yang sudah dijelaskan sebelumnya, *slice* dapat dibentuk dari array yang sudah didefinisikan, dengan teknik 2 indeks untuk mengambil elemen-elemennya. Berikut adalah contoh pembentukan *slice* dari array.

```
1    var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop", "web"}
2    var kursus_saya = kursus[0:3]
3
4    fmt.Println(kursus_saya) //[dbms, erp, server-os]
```

Berdasarkan kode diatas, variabel baru bernama **kursus_saya** akan mengambil isi dari variabel **kursus** yang dimulai dari indeks ke-0 hingga elemen sebelum indeks ke-3. Berikut beberapa contoh dari pembentukan *slice*.

Tabel 4.1 Contoh Pembentukan Slice

Kode	Output	Penjelesan
kursus[0:2]	[dbms, erp]	Dimulai dari elemen indeks ke-0 sampai sebelum indeks ke-1
kursus[0:6]	[dbms, erp, server-os, networking, desktop, web]	Diulai dari elemen indeks ke-0 sampai sebelum indeks ke-6
kursus[0:0]	[]	Dimulai dari elemen indeks ke-0 sampai sebelum indeks ke-0, akan menghasil kan *slice* kosong
kursus[6:6]	[]	Dimulai dari elemen indeks ke-6, akan mengahasilkan *slice* kosong
kursus[:]	[dbms, erp, server-os, networking, desktop, web]	Mengambil seluruh elemen
kursus[2:]	[server-os, networking, dekstop, web]	Mengambil seluruh elemen dimulai dari indeks ke-2
kursus[:2]	[dbms, erp]	Mengambil seluruh elemen sampai elemen sebelum indeks ke-2

Seperti dijelaskan sebelumnya bahwa *slice* merupakan tipe data *reference* atau referensi, jika suatu *slice* dibentuk dari *slice* lainnya maka data elemen *slice* yang baru akan memiliki alamat memori yang sama dengan *slice* yang lama. Perubahan pada *slice* baru tersebut akan mempengaruhi *slice* yang lama, untuk lebih jelasnya dapat dilihat pada contoh di bawah.

```
1     var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop", "web"}
2
3     var kursus_saya = kursus[0:2]
4     fmt.Println(kursus) //[dbms, erp, server-os, networking, desktop, web]
5     fmt.Println(kursus_saya) // [dbms, erp]
6
7     kursus_saya[0] = "database"
8
9     fmt.Println(kursus) //[database, erp, server-os, networking, desktop, web]
10    fmt.Println(kursus_saya) //[database, erp]
```

Selain pembentukan *slice* dengan teknik 2 indeks, pembentukan *slice* dapat dilakukan dengan teknik 3 indeks. Teknik ini digunakan ketika ingin menentukan kapasitas dari *slice* yang baru ketika dibentuk, namun nilai dari kapasitas *slice* yang akan dibentuk tidak boleh melebihi

kapasitas *slice* yang akan di slicing. Berikut contoh penggunaan dari *slice* dengan teknik 3 indeks.

```
var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop", "web"}

var kursus_saya = kursus[0:3:3]

fmt.Println(kursus_saya) //[dbms erp server-os]
fmt.Println(cap(kursus_saya)) //3
```

Membuat *Slice* Dengan *Make*

Selain cara diatas, suatu *slice* dapat juga dibentuk dengan menggunakan *keyword make*, pembuatan *slice* dengan *keyword make* ini akan membuat suatu array yang dinamis. Pada contoh dibawah mungkin tidak akan terlihat perbedaanya dengan array pada umumnya, namun akan terlihat ketika kode yang akan digunakan lebih kompleks.

```
var kursus_urang = make([]string, 5)

fmt.Println(kursus_urang)// [    ]
```

Fungsi Dalam *Slice*

Seperti hal nya dalam array, terdapat fungsi-fungsi dasar yang dapat dioperasi kan dalam penggunaan *slice*, yaitu sebagai berikut :

1. Fungsi *len()*

Fungsi *len()* digunakan untuk menghitung jumlah elemen dari suatu *slice*. Berikut adalah contoh dari fungsi *len()*.

```
var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop", "web"}

fmt.Println(len(kursus)) //6
```

2. Fungsi *cap()*

Fungsi *cap()* digunakan untuk menghitung kapasitas maksimum yang tersedia dalam suatu *slice*, nilai fungsi *cap()* akan memiliki nilai seperti fungsi *len()* ketika pertama dibuat, namun dapat berubah seiring terjadinya operasi *slice*. Berikut adalah contoh dari fungsi *cap()*.

```go
var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop", "web"}

fmt.Println(len(kursus)) //6
fmt.Println(cap(kursus)) //6

var kursus_saya = kursus[0:3]

fmt.Println(len(kursus_saya)) //3
fmt.Println(cap(kursus_saya)) //6

var kursus_abdi = kursus[1:3]

fmt.Println(len(kursus_abdi)) //2
fmt.Println(cap(kursus_abdi)) //5
```

Dapat dilihat diatas pada saaat pembuatan *slice* pertama kali akan menghasilkan nilai *cap()* sama seperti nilai *len()*, lalu saat membuat *slice* baru dengan indeks [0:3] atau dimulai dari indeks ke-0 sampai indeks sebelum ke-3, nilai cap tetap 6 dikarenakan pada saat pengambilan indeks dimulai dari 0 tidak terjadi perubahan pada indeks pertamanya, lain hal dengan pembuatan *slice* berikutnya dengan indeks [1:3] atau dimulai dari indeks ke-1 sampai indeks sebelum ke-3, nilai cap berubah menjadi 5 dikarenakan saat pengambilan indeks dimulai dari indeks ke-1, maka terjadi perubahan pada indeks pertamanya yang berarti indeks ke-0 sudah dihilangkan, oleh karena itu nilai kapasitasnya berkurang.

> Nilai kosong dari suatu *slice* adalah nil. Nilai len() dan cap() dari *slice* nil adalah 0.

Fungsi *Append()*

Fungsi *append()* digunakan untuk menambahkan elemen pada *slice*. Elemen yang baru ditambahkan akan di posisikan setelah indeks paling akhir. Dan nilai balik dari *append()* adalah *slice* yang sudah ditambahkan nilai baru. Berikut adalah contoh penggunaan *append()*.

```go
var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop", "web"}

var kursus_saya = kursus[1:3]

kursus_saya = append(kursus_saya, "manajemen-tik")
fmt.Println(len(kursus_saya)) // 3
fmt.Println(cap(kursus_saya)) // 4
fmt.Println(kursus_saya) //[erp server-os manajemen-tik]
fmt.Println(kursus)//[dbms erp server-os manajemen-tik desktop web]

```

Dalam penggunaan *append()* terdapat beberapa yang harus diperhatikan, yaitu:

a. *Append()* akan membuat referensi baru ketika nilai dari kapasitas (cap) sama dengan jumlah elemen (len).
b. *Append()* akan mengisi kapasitas yang tersedia ketika nilai kapasitas (cap) lebih besar dari pada (len), dan akan merubah nilai referensi *slice* yang berhubungan.

Berdasarkan contoh gambar ditatas dapat diperhatikan ketika *slice* **kursus_saya** melakukan operasi *append()* dengan nilai data yang dimasukan adalah **"manajemen-tik",** dikarenakan nilai kapasitas lebih besar dari pada jumlah elemen, maka elemen yang ditambah akan memasuki elemen yang kosong, yaitu elemen terakhir atau indeks ke-3, seperti penjelasan diatas ketika melakukan *append()* maka nilai referensi *slice* yang berhubungan dengan *slice* tersebut akan ikut berubah, dapat dilihat pada gambar diatas baris ke 9, bahwa *slice* **kursus** merubah indeks ke-3 (networking) dengan manajemen tik.

3. Fungsi *Copy()*

Fungsi *copy()* digunakan untuk menduplikasi elemen *slice* pada *parameter* ke-2 untuk *parameter* ke-1, nilai balik dari fungsi *copy()* adalah jumlah nilai yang berhasil di duplikasi. Berikut contoh penggunaan fungsi copy().

```
1     var kursus =[]string{"dbms", "erp", "server-os", "networking", "desktop"}
2
3
4     var kursus_beginner = []string{"oracle","nav", "linux", "cisco", "java", "golang"}
5
6
7     var kursus_saya = copy(kursus, kursus_beginner)
8
9     fmt.Println(kursus_saya) //5
10    fmt.Println(kursus) //[oracle nav linux cisco java]
11    fmt.Println(kursus_beginner) //[oracle nav linux cisco java golang]
12
13
```

Pada contoh diatas *slice* **kursus_beginner** akan diduplikasi untuk *slice* **kursus**. Nilai kapasitas dari *slice* **kursus** adalah 5 elemen, dan nilai kapasitas dari *slice* **kursus_beginner** adalah 6 elemen, ketika dilakukan fungsi copy() dari parameter ke-2 **(kursus_beginner)** untuk parameter ke-1 (**kursus**) akan menghasilkan nilai balik 5, dikarenan jumlah dari kapasitas *slice* **kursus** hanya 5 walaupun nilai kapasitas yang diduplikasi dari **kursus_beginner** adalah 6.

4.4 Map

Pengenalan *Map*

Map adalah tipe data asosiatif yang ada di Golang, berbentuk *key-value* . Untuk setiap data (atau *value*) yang disimpan, disiapkan juga *key*-nya. *Key* memiliki sifat unik, karena digunakan sebagai penanda (atau identifier) untuk pengaksesan *value* yang bersangkutan. Kalau dilihat, *map* mirip seperti *slice*, hanya saja indeks yang digunakan untuk pengaksesan bisa ditentukan sendiri tipe-nya (indeks tersebut adalah *key*)

Inisialisasi *Map*

Pembentukan *Map* dapat dilakukan dengan *keyword map* serta diikuti dengan tipe data *key* dan *value* -nya. Berikut adalah contoh pembentukan suatu *map* .

```
1 var sesi = map[int]string{}
2
3 sesi[1] = "dbms"
4
5 fmt.Println(sesi[1]) //dbms
6 fmt.Println(sesi[2]) //   => tanpa nilai
```

Pada contoh diatas dibentuk suatu *map* dengan tipe data *key*-nya adalah *integer* dan tipe data *value* -nya adalah *string*. Pada dasarnya nilai suatu *map* adalah nil atau kosong, lalu untuk melakukan penge-set-an pada suatu *map* dapat dilakukan dengan menggunakan *key* yang diikuti kurang kotak berserta *value* yang ingin dimasukan, seperti contoh pada gambar diatas baris ke 3. Pengambilan *value* pada suatu *map* dapat dilakukan hanya dengan *keyword map* diikuti dengan *key* dari *value* yang ingin diambil, pengambilan *value* yang dimana *key*-nya tidak terdaftar dalam *map* akan memberikan nilai *default* dari tipe data *value* tersebut. Pada contoh gambar diatas baris ke 6, *map* sesi tidak memiliki *key* 0 sehingga ketika *key* 0 dipanggil akan mengeluarkan *string* kosong, hal ini berdasarkan nilai default dari tipe data *string*.

Terdapat cara lain untuk melakukan penge-set-an suatu nilai *map*, yaitu dengan cara memasukan semua nilai yang ingin dimasukan dalam kurung kurawal, seperti contoh pada gambar dibawah ini.

```
    var sesi = map[int]string{
        1 : "dbms",
        2 : "server-os",
    }


    fmt.Println(sesi[1]) //dbms
    fmt.Println(sesi[2]) //server-os
```

Key dan *value* dituliskan dengan pembatas tanda titik dua (**:**). Sedangkan tiap itemnya dituliskan dengan pembatas tanda koma (**,**).

Tipe data *value* dari suatu *map* tidak hanya dapat digunakan untuk tipe data primitive saja namun dapat pula digunakan pada tipe data seperti struct dan *slice*, berikut adalah contoh penggunaannya.

- **Penggunaan *Map* dengan Struct**

```
package main

import "fmt"

type Kursus struct {
    sesi int
    materi string
}

func main() {
    var mahasiswa = map[string]Kursus{
        "Aria" : Kursus{
            1,"dbms",
        },
    }

  fmt.Println(mahasiswa["Aria"]) // {1 dbms}
  fmt.Println(mahasiswa["Aria"].materi) //dbms
}
```

Bagian ini tidak akan dijelaskan kembali cara membuat suatu struct karena sudah dibahas pada bagian sebelumnya. Pada contoh diatas *map* dibuat dengan tipe data key berupa string dan tipe data *value* nya adalah sebuah struct **Kursus** berisi variabel sesi ber tipe data integer dan variabel materi ber tipe data string.

- **Penggunaan *Slice* dengan *Map***

```
1     var kursus = []map[string]string{
2         map[string]string{"sesi" : "1" , "kursus" : "dbms"},
3         map[string]string{"sesi" : "2" , "kursus" : "server-os"},
4         map[string]string{"sesi" : "3" , "kursus" : "networking"},
5     }
6
7     fmt.Println("saya kursus " + kursus[0]["kursus"] + " sesi " + kursus[1]["sesi"])
8     //saya kursus dbms sesi 1
9
10    for _, kursus_saya := range kursus {
11        fmt.Println(kursus_saya["kursus"])
12        //dbms
13        //server-os
14        //networking
15    }
```

Pada contoh diatas variabel **Kursus** memiliki tipe data *slice* yang dimana memiliki *value* ber tipe data *map*. Untuk inisialisasi setiap *value* dari *slice* dilakukan seperti inisialisasi menggunakan *map*, dapat dilihat pada baris 2, 3, dan 4. Setiap inisialisasi yang dilakukan sama seperti insialisasi pada *map*, lalu untuk megambil data dari variabel **Kursus** tersebut dapat dilakukan dengan cara mengambil indeks ke-berapa dari *slice* tersebut lalu diikuti key dari *map* tersebut, seperti contoh pada baris ke-7 dilakukan pengambilan data pada indeks *slice* ke-0 dan diikut dengan mengambil *value* dari key kursus dan key sesi, selain itu dapat pula diambil datanya menggunakan for := range yang sudah dibahas pada bagian perulangan sebelumnya.

Menghapus *Item Map*

Untuk menghapus suatu item dalam variabel *map* dapat menggunakan fungsi *delete()*. Penggunaan fungsi ini dapat digunakan dengan memasukan objek *map* dan key item yang ingin dihapus sebagai parameter fungsi ini. Berikut adalah contoh penggunaan fungsi *delete()*.

```
1     var sesi = map[int]string{
2         1 : "dbms",
3         2 : "server-os",
4     }
5
6     fmt.Println(sesi) // map[2:server-os 1:dbms]
7     delete(sesi, 1) //menghapus map sesi dengan key 1
8     fmt.Println(sesi) // map[2:server-os]
```

Menguji Keberadan Suatu *Item Key* Dalam *Map*

Pada pembahasan sebelumnya, jika suatu key yang tidak terdapat dalam *map* diakses maka nilai yang akan dikeluarkan adalah nilai default dari tipe data *value* variabel *map* tersebut. Untuk mencegah mengeluarkan nilai default, golang menyediakan suatu fungsi untuk

mengatahui apakah dalam sebuah variabel *map* tersebut terdapat item dengan key tertentu, yaitu dengan memanfaatkan 2 variabel untuk menampung nilai pengaksesan item. Variabel ke-2 akan berisikan nilai bool yang menunjukan ada tau tidak item yang dicari. Berikut adalah contoh

```
    var sesi = map[int]string{
        1 : "dbms",
        2 : "server-os",
    }

    var nilai, status = sesi[1] //value, bool

    if status{
        fmt.Println(nilai)
      //jika status true print nilai
    }else {
        fmt.Println("kursus tidak ditemukan")
      // jika false print "kursus tidak ditemukan"
    }
```

Pada contoh diatas variabel nilai bertugas menampung isi dari *map* sesi dengan key 1, dan variabel status akan terisi dengan bool(true) jika ditemukan key tersebut dan bool(false) jika tidak ditemukan.