MODUL VI
ARRAY, SLICE ARRAY & MAP ARRAY

Pertemuan : 6
Waktu : 8 x 60 menit (Online)

1.1 Tujuan Modul III

Setelah mahasiswa mempelajari materi ini, diharapkan dapat :

1. Memahami pemakaian array di golang.
2. Memahami slicing array di golang.
3. Memahami map array di golang.

1.2 Landasan Teori

1.2.1 Array

Array adalah kumpulan data bertipe sama, yang disimpan dalam sebuah variabel. Array memiliki kapasitas yang nilainya ditentukan pada saat pembuatan, menjadikan elemen/data yang disimpan di array tersebut jumlahnya tidak boleh melebihi yang sudah dialokasikan. Default nilai tiap elemen array pada awalnya tergantung dari tipe datanya. Jika `int` maka tiap element zero value-nya adalah `0`, jika `bool` maka `false`, dan seterusnya. Setiap elemen array memiliki indeks berupa angka yang merepresentasikan posisi urutan elemen tersebut. Indeks array dimulai dari 0.

Contoh penerapan array:

```
var names [4]string
names[0] = "trafalgar"
names[1] = "d"
names[2] = "water"
names[3] = "law"

fmt.Println(names[0], names[1], names[2], names[3])
```

Variabel `names` dideklarasikan sebagai `array string` dengan alokasi elemen `4` slot. Cara mengisi slot elemen array bisa dilihat di kode di atas, yaitu dengan langsung mengakses elemen menggunakan indeks, lalu mengisinya.

1.2.1.1 Pengisian Elemen Array yang Melebihi Alokasi Awal

Pengisian elemen array pada indeks yang tidak sesuai dengan alokasi menghasilkan error. Contoh sederhana, jika array memiliki 4 slot, maka pengisian nilai slot 5 seterusnya adalah tidak valid.

```
var names [4]string
names[0] = "trafalgar"
names[1] = "d"
names[2] = "water"
names[3] = "law"
names[4] = "ez" // baris kode ini menghasilkan error
```

Solusi dari masalah di atas adalah dengan menggunakan keyword `append`, yang di bab selanjutnya (Bab A. Slice) akan kita bahas.

1.2.1.2 Inisialisasi Nilai Awal Array

Pengisian elemen array bisa dilakukan pada saat deklarasi variabel. Caranya dengan menuliskan data elemen dalam kurung kurawal setelah tipe data, dengan pembatas antar elemen adalah tanda koma (`,`).

```
var fruits = [4]string{"apple", "grape", "banana", "melon"}


fmt.Println("Jumlah element \t\t", len(fruits))

fmt.Println("Isi semua element \t", fruits)
```

Penggunaan fungsi `fmt.Println()` pada data array tanpa mengakses indeks tertentu, akan menghasilkan output dalam bentuk string dari semua array yang ada. Teknik ini biasa digunakan untuk **debugging** data array.

1.2.1.3 Inisialisasi Nilai Array Dengan Gaya Vertikal

Elemen array bisa dituliskan dalam bentuk horizontal (seperti yang sudah dicontohkan di atas) ataupun dalam bentuk vertikal.

```
var fruits [4]string

// cara horizontal
fruits  = [4]string{"apple", "grape", "banana", "melon"}

// cara vertikal
fruits  = [4]string{
    "apple",
    "grape",
    "banana",
    "melon",
}
```

Khusus untuk deklarasi array dengan cara vertikal, tanda koma wajib dituliskan setelah elemen, termasuk elemen terakhir. Jika tidak, maka akan muncul error.

1.2.1.4 Inisialisasi Nilai Awal Array Tanpa Jumlah Elemen

Deklarasi array yang nilainya di set di awal, boleh tidak dituliskan jumlah lebar array-nya, cukup ganti dengan tanda 3 titik (`...`). Jumlah elemen akan dikalkulasi secara otomatis menyesuaikan data elemen yang diisikan.

```
var numbers = [...]int{2, 3, 2, 4, 3}

fmt.Println("data array \t:", numbers)
fmt.Println("jumlah elemen \t:", len(numbers))
```

Variabel `numbers` akan secara otomatis memiliki jumlah elemen `5`, karena pada saat deklarasi disiapkan 5 buah elemen.

1.2.1.5 Array Multidimensi

Array multidimensi adalah array yang tiap elemennya juga berupa array (dan bisa seterusnya, tergantung kedalaman dimensinya).Cara deklarasi array multidimensi secara umum sama dengan cara deklarasi array biasa, dengan cara menuliskan data array dimensi selanjutnya sebagai elemen array dimensi sebelumnya.Khusus untuk array yang merupakan sub dimensi atau elemen, boleh tidak dituliskan jumlah datanya. Contohnya bisa dilihat pada deklarasi variabel `numbers2` di kode berikut.

```
var numbers1 = [2][3]int{[3]int{3, 2, 3}, [3]int{3, 4, 5}}
var numbers2 = [2][3]int{{3, 2, 3}, {3, 4, 5}}

fmt.Println("numbers1", numbers1)
fmt.Println("numbers2", numbers2)
```

1.2.1.6 Perulangan Elemen Array Menggunakan Keyword for

Keyword `for` dan array memiliki hubungan yang sangat erat. Dengan memanfaatkan perulangan menggunakan keyword ini, elemen-elemen dalam array bisa didapat.

Ada beberapa cara yang bisa digunakan untuk melooping data array, yg pertama adalah dengan memanfaatkan variabel iterasi perulangan untuk mengakses elemen berdasarkan indeks-nya. Contoh:

```
var fruits = [4]string{"apple", "grape", "banana",
"melon"}

for i := 0; i < len(fruits); i++ {
    fmt.Printf("elemen %d : %s\n", i, fruits[i])
}
```

Perulangan di atas dijalankan sebanyak jumlah elemen array `fruits` (bisa diketahui dari kondisi `i < len(fruits)`). Di tiap perulangan, elemen array diakses lewat variabel iterasi `i`.

1.2.1.7 Perulangan Elemen Array Menggunakan Keyword for - range

Ada cara yang lebih sederhana me-looping data array, dengan menggunakan keyword `for` - `range`. Contoh pengaplikasiannya bisa dilihat di kode berikut.

```
var fruits = [4]string{"apple", "grape", "banana", "melon"}

for i, fruit := range fruits {
    fmt.Printf("elemen %d : %s\n", i, fruit)
}
```

Array `fruits` diambil elemenya secara berurutan. Nilai tiap elemen ditampung variabel oleh `fruit` (tanpa huruf s), sedangkan indeks nya ditampung variabel `i`.

Output program di atas, sama dengan output program sebelumnya, hanya cara yang digunakan berbeda.

1.2.1.8 Penggunaan Variabel Underscore _ Dalam for - range

Kadang kala ketika *looping* menggunakan `for` - `range`, ada kemungkinan dimana data yang dibutuhkan adalah elemen-nya saja, indeks-nya tidak. Sedangkan kode di atas, `range` mengembalikan 2 data, yaitu indeks dan elemen.

Seperti yang sudah diketahui, bahwa di Go tidak memperbolehkan adanya variabel yang menganggur atau tidak dipakai. Jika dipaksakan, error akan muncul, contohnya seperti kode berikut.

```
var fruits = [4]string{"apple", "grape", "banana", "melon"}

for i, fruit := range fruits {
    fmt.Printf("nama buah : %s\n", fruit)
}
```

Hasil dari kode program di atas:

```
novalagung:belajar-golang $ go run bab14.go
# command-line-arguments
./bab14.go:8: i declared and not used
novalagung:belajar-golang $
```

Disinilah salah satu kegunaan variabel pengangguran, atau underscore (_). Tampung saja nilai yang tidak ingin digunakan ke underscore.

```
var fruits = [4]string{"apple", "grape", "banana", "melon"}

for _, fruit := range fruits {
    fmt.Printf("nama buah : %s\n", fruit)
}
```

Pada kode di atas, yang sebelumnya adalah variabel `i` diganti dengan _, karena kebetulan variabel `i` tidak digunakan.Jika yang dibutuhkan hanya indeks elemen-nya saja, bisa gunakan 1 buah variabel setelah keyword `for`.

1.2.1.9 Alokasi Elemen Array Menggunakan Keyword make

Deklarasi sekaligus alokasi data array juga bisa dilakukan lewat keyword `make`.

```
var fruits = make([]string, 2)
fruits[0] = "apple"
fruits[1] = "manggo"

fmt.Println(fruits)  // [apple manggo]
```

Parameter pertama keyword `make` diisi dengan tipe data elemen array yang diinginkan, parameter kedua adalah jumlah elemennya. Pada kode di atas, variabel `fruits` tercetak sebagai array string dengan alokasi 2 slot.

1.2.2 Array Slice

Slice adalah *reference* elemen array. Slice bisa dibuat, atau bisa juga dihasilkan dari manipulasi sebuah array ataupun slice lainnya. Karena merupakan data *reference*, menjadikan perubahan data di tiap elemen slice akan berdampak pada slice lain yang memiliki alamat memori yang sama.

1.2.2.1 Inisialisasi Slice

Cara pembuatan slice mirip seperti pembuatan array, bedanya tidak perlu mendefinisikan jumlah elemen ketika awal deklarasi. Pengaksesan nilai elemen-nya juga sama. Contoh pembuatan slice:

```
var fruits = []string{"apple", "grape", "banana", "melon"}
fmt.Println(fruits[0]) // "apple"
```

Salah satu perbedaan slice dan array bisa diketahui pada saat deklarasi variabel-nya, jika jumlah elemen tidak dituliskan, maka variabel tersebut adalah slice.

```
var fruitsA = []string{"apple", "grape"}      // slice
var fruitsB = [2]string{"banana", "melon"}    // array
var fruitsC = [...]string{"papaya", "grape"}  // array
```

1.2.2.2 Hubungan Slice Dengan Array & Operasi Slice

Kalau perbedannya hanya di penentuan alokasi pada saat inisialisasi, kenapa tidak menggunakan satu istilah saja? atau adakah perbedaan lainnya?

Sebenarnya slice dan array tidak bisa dibedakan karena merupakan sebuah kesatuan. Array adalah kumpulan nilai atau elemen, sedang slice adalah referensi tiap elemen tersebut.

Slice bisa dibentuk dari array yang sudah didefinisikan, caranya dengan memanfaatkan teknik **2 index** untuk mengambil elemen-nya. Contoh bisa dilihat pada kode berikut.

```
var fruits = []string{"apple", "grape", "banana", "melon"}
var newFruits = fruits[0:2]

fmt.Println(newFruits) // ["apple", "grape"]
```

Kode `fruits[0:2]` maksudnya adalah pengaksesan elemen dalam slice `fruits` yang **dimulai dari indeks ke-0, hingga elemen sebelum indeks ke-2**. Elemen yang memenuhi kriteria tersebut akan didapat, untuk kemudian disimpan pada variabel lain sebagai slice baru. Pada contoh di atas, `newFruits` adalah slice baru yang tercetak dari slice `fruits`, dengan isi 2 elemen, yaitu `"apple"` dan `"grape"`.

Ketika mengakses elemen array menggunakan satu buah indeks (seperti `data[2]`), nilai yang didapat merupakan hasil **copy** dari referensi aslinya. Berbeda dengan pengaksesan elemen menggunakan 2 indeks

(seperti `data[0:2]`), nilai yang didapat adalah *reference* elemen atau slice.

Tabel berikut adalah list operasi operasi menggunakan teknik 2 indeks yang bisa dilakukan.

```
var fruits = []string{"apple", "grape", "banana", "melon"}
```

Kode	Output	Penjelasan
fruits[0:2]	[apple, grape]	semua elemen mulai indeks ke-0, hingga sebelum indeks ke-2
fruits[0:4]	[apple, grape, banana, melon]	semua elemen mulai indeks ke-0, hingga sebelum indeks ke-4
fruits[0:0]	[ ]	menghasilkan slice kosong, karena tidak ada elemen sebelum indeks ke-0
fruits[4:4]	[ ]	menghasilkan slice kosong, karena tidak ada elemen yang dimulai dari indeks ke-4
fruits[4:0]	[ ]	error, pada penulisan fruits[a,b] nilai a harus lebih besar atau sama dengan b
fruits[:]	[apple, grape, banana, melon]	semua elemen
fruits[2:]	[banana, melon]	semua elemen mulai indeks ke-2
fruits[:2]	[apple, grape]	semua elemen hingga sebelum indeks ke-2

1.2.2.3 Slice Merupakan Tipe Data Reference

Slice merupakan tipe data *reference* atau referensi. Artinya jika ada slice baru yang terbentuk dari slice lama, maka data elemen slice yang baru akan memiliki alamat memori yang sama dengan elemen slice lama. Setiap perubahan yang terjadi di elemen slice baru, akan berdampak juga pada elemen slice lama yang memiliki referensi yang sama.

Program berikut merupakan pembuktian tentang teori yang baru kita bahas. Kita akan mencoba mengubah data elemen slice baru, yang terbentuk dari slice lama.

```
var fruits = []string{"apple", "grape", "banana", "melon"}

var aFruits = fruits[0:3]
var bFruits = fruits[1:4]

var aaFruits = aFruits[1:2]
var baFruits = bFruits[0:1]

fmt.Println(fruits)   // [apple grape banana melon]
fmt.Println(aFruits)  // [apple grape banana]
fmt.Println(bFruits)  // [grape banana melon]
fmt.Println(aaFruits) // [grape]
fmt.Println(baFruits) // [grape]

// Buah "grape" diubah menjadi "pineapple"
baFruits[0] = "pineapple"

fmt.Println(fruits)   // [apple pineapple banana melon]
fmt.Println(aFruits)  // [apple pineapple banana]
fmt.Println(bFruits)  // [pineapple banana melon]
fmt.Println(aaFruits) // [pineapple]
fmt.Println(baFruits) // [pineapple]
```

Sekilas bisa kita lihat bahwa setelah slice yang isi datanya adalah `grape` diubah menjadi `pineapple`, semua slice pada 4 variabel lainnya juga ikut berubah.

Variabel `aFruits`, `bFruits` merupakan slice baru yang terbentuk dari variabel `fruits`. Dengan menggunakan dua slice baru tersebut, diciptakan lagi slice lainnya, yaitu `aaFruits`, dan `baFruits`. Kelima slice tersebut ditampilkan nilainya.

Selanjutnya, nilai dari `baFruits[0]` diubah, dan 5 slice tadi ditampilkan lagi. Hasilnya akan ada banyak slice yang elemennya ikut berubah. Yaitu elemen-elemen yang referensi-nya sama dengan referensi element `baFruits[0]`.

1.2.2.4 Fungsi len()

Fungsi `len()` digunakan untuk menghitung jumlah elemen slice yang ada. Sebagai contoh jika sebuah variabel adalah slice dengan data 4 buah, maka fungsi ini pada variabel tersebut akan mengembalikan angka **4**.

```
var fruits = []string{"apple", "grape", "banana", "melon"}
fmt.Println(len(fruits)) // 4
```

1.2.2.5 Fungsi cap()

Fungsi `cap()` digunakan untuk menghitung lebar atau kapasitas maksimum slice. Nilai kembalian fungsi ini untuk slice yang baru dibuat pasti sama dengan `len`, tapi bisa berubah seiring operasi slice yang dilakukan. Agar lebih jelas, silakan disimak kode berikut.

```
var fruits = []string{"apple", "grape", "banana", "melon"}
fmt.Println(len(fruits))  // len: 4
fmt.Println(cap(fruits))  // cap: 4

var aFruits = fruits[0:3]
fmt.Println(len(aFruits)) // len: 3
fmt.Println(cap(aFruits)) // cap: 4

var bFruits = fruits[1:4]
fmt.Println(len(bFruits)) // len: 3
fmt.Println(cap(bFruits)) // cap: 3
```

Variabel `fruits` disiapkan di awal dengan jumlah elemen 4, fungsi `len(fruits)` dan `cap(fruits)` pasti hasinya 4.

Variabel `aFruits` dan `bFruits` merupakan slice baru berisikan 3 buah elemen milik slice `fruits`. Variabel `aFruits` mengambil elemen index 0, 1, 2; sedangkan `bFruits` 1, 2, 3.

Fungsi `len()` menghasilkan angka 3, karena jumlah elemen kedua slice ini adalah 3. Tetapi `cap(aFruits)` menghasilkan angka yang

berbeda, yaitu 4 untuk `aFruits` dan 3 untuk `bFruits`. kenapa? jawabannya bisa dilihat pada tabel berikut.

Kode	**Output**	**len()**	**cap()**
`fruits[0:4]`	**[buah buah buah buah]**	4	4
`aFruits[0:3]`	**[buah buah buah** `----`**]**	3	4
`bFruits[1:3]`	`----` **[buah buah buah]**	3	3

Kita analogikan slicing 2 index menggunakan **x** dan **y**.

```
fruits[x:y]
```

Slicing yang dimulai dari indeks **0** hingga **y** akan mengembalikan elemen-elemen mulai indeks **0** hingga sebelum indeks **y**, dengan lebar kapasitas adalah sama dengan slice aslinya.

Sedangkan slicing yang dimulai dari indeks **x**, yang dimana nilai **x** adalah lebih dari **0**, membuat elemen ke-**x** slice yang diambil menjadi elemen ke-0 slice baru. Hal inilah yang membuat kapasitas slice berubah.

1.2.2.6 Fungsi append()

Fungsi `append()` digunakan untuk menambahkan elemen pada slice. Elemen baru tersebut diposisikan setelah indeks paling akhir. Nilai balik fungsi ini adalah slice yang sudah ditambahkan nilai barunya. Contoh penggunaannya bisa dilihat di kode berikut.

```
var fruits = []string{"apple", "grape", "banana"}
var cFruits = append(fruits, "papaya")

fmt.Println(fruits)  // ["apple", "grape", "banana"]
fmt.Println(cFruits) // ["apple", "grape", "banana", "papaya"]
```

Ada 2 hal yang perlu diketahui dalam penggunaan fungsi ini.

- Ketika jumlah elemen dan lebar kapasitas adalah sama (`len(fruits) == cap(fruits)`), maka elemen baru hasil `append()` merupakan referensi baru.
- Ketika jumlah elemen lebih kecil dibanding kapasitas (`len(fruits) < cap(fruits)`), elemen baru tersebut ditempatkan kedalam cakupan kapasitas, menjadikan semua elemen slice lain yang referensi-nya sama akan berubah nilainya.

Agar lebih jelas silahkan perhatikan contoh berikut.

```
var fruits = []string{"apple", "grape", "banana"}
var bFruits = fruits[0:2]

fmt.Println(cap(bFruits)) // 3
fmt.Println(len(bFruits)) // 2

fmt.Println(fruits)  // ["apple", "grape", "banana"]
fmt.Println(bFruits) // ["apple", "grape"]

var cFruits = append(bFruits, "papaya")

fmt.Println(fruits)  // ["apple", "grape", "papaya"]
fmt.Println(bFruits) // ["apple", "grape"]
fmt.Println(cFruits) // ["apple", "grape", "papaya"]
```

Pada contoh di atas bisa dilihat, elemen indeks ke-2 slice `fruits` nilainya berubah setelah ada penggunaan keyword `append()` pada `bFruits`. Slice `bFruits` kapasitasnya adalah **3** sedang jumlah datanya hanya **2**. Karena `len(bFruits) < cap(bFruits)`, maka elemen baru yang dihasilkan, terdeteksi sebagai perubahan nilai pada referensi yang lama (referensi elemen indeks ke-2 slice `fruits`), membuat elemen yang referensinya sama, nilainya berubah.

1.2.2.7 Fungsi copy()

Fungsi `copy()` digunakan untuk men-copy elements slice pada `src` (parameter ke-2), ke `dst` (parameter pertama).

```
copy(dst, src)
```

Jumlah element yang di-copy dari `src` adalah sejumlah lebar slice `dst` (atau `len(dst)`). Jika jumlah slice pada `src` lebih kecil dari `dst`, maka akan tercopy semua. Lebih jelasnya silahkan perhatikan contoh berikut.

```
dst := make([]string, 3)
src := []string{"watermelon", "pineapple", "apple",
"orange"}
n := copy(dst, src)

fmt.Println(dst) // watermelon pineapple apple
fmt.Println(src) // watermelon pineapple apple orange
fmt.Println(n)   // 3
```

Pada kode di atas variabel slice `dst` dipersiapkan dengan lebar adalah 3 elements. Slice `src` yang isinya 4 elements, di-copy ke `dst`. Menjadikan isi slice `dst` sekarang adalah 3 buah elements yang sama dengan 3 buah elements `src`, hasil dari operasi `copy()`.

`Fungsi copy() mengembalikan informasi angka, representasi dari jumlah elemen yang berhasil di-copy.`

Yang tercopy hanya 3 buah (meski `src` memiliki 4 elements) hal ini karena `copy()` hanya meng-copy elements sebanyak `len(dst)`.

Pada contoh kedua berikut, `dst` merupakan slice yang sudah ada isinya, 3 buah elements. Variabel `src` yang juga merupakan slice dengan isi dua elements, di-copy ke `dst`. Karena operasi `copy()` akan meng-copy sejumlah `len(dst)`, maka semua elements `src` akan tercopy **karena jumlahnya dibawah atau sama dengan lebar** `dst`.

```
dst := []string{"potato", "potato", "potato"}
src := []string{"watermelon", "pinnaple"}
n := copy(dst, src)

fmt.Println(dst) // watermelon pinnaple potato
fmt.Println(src) // watermelon pinnaple
fmt.Println(n)   // 2
```

Jika dilihat pada kode di atas, isi `dst` masih tetap 3 elements, tapi dua elements pertama adalah sama dengan `src`. Element terakhir `dst` isinya tidak berubah, tetap `potato`, hal ini karena proses copy hanya memutasi element ke-1 dan ke-2 milik `dst`, karena memang pada `src` hanya dua itu elements-nya.

1.2.2.8 Pengaksesan Elemen Slice Dengan 3 Indeks

3 index adalah teknik slicing elemen yang sekaligus menentukan kapasitasnya. Cara menggunakannya yaitu dengan menyisipkan angka kapasitas di belakang, seperti `fruits[0:1:1]`. Angka kapasitas yang diisikan tidak boleh melebihi kapasitas slice yang akan di slicing.

Berikut merupakan contoh penerapannya.

```
var fruits = []string{"apple", "grape", "banana"}
var aFruits = fruits[0:2]
var bFruits = fruits[0:2:2]

fmt.Println(fruits)       // ["apple", "grape", "banana"]
fmt.Println(len(fruits)) // len: 3
fmt.Println(cap(fruits)) // cap: 3

fmt.Println(aFruits)      // ["apple", "grape"]
fmt.Println(len(aFruits)) // len: 2
fmt.Println(cap(aFruits)) // cap: 3

fmt.Println(bFruits)      // ["apple", "grape"]
fmt.Println(len(bFruits)) // len: 2
fmt.Println(cap(bFruits)) // cap: 2
```

1.2.3 Array Map

Map adalah tipe data asosiatif yang ada di Go, berbentuk *key-value pair*. Untuk setiap data (atau value) yang disimpan, disiapkan juga key-nya. Key harus 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).

1.2.3.1 Penggunaan Map

Cara menggunakan map cukup dengan menuliskan keyword `map` diikuti tipe data key dan value-nya. Agar lebih mudah dipahami, silakan perhatikan contoh di bawah ini.

```
var chicken map[string]int
chicken = map[string]int{}

chicken["januari"] = 50
chicken["februari"] = 40

fmt.Println("januari", chicken["januari"]) // januari 50
fmt.Println("mei",     chicken["mei"])     // mei 0
```

Variabel `chicken` dideklarasikan sebagai map, dengan tipe data key adalah `string` dan value-nya `int`. Dari kode tersebut bisa dilihat bagaimana cara penggunaan keyword `map`.

Kode `map[string]int` maknanya adalah, tipe data `map` dengan key bertipe `string` dan value bertipe `int`.

Default nilai variabel `map` adalah `nil`. Oleh karena itu perlu dilakukan inisialisasi nilai default di awal, caranya cukup dengan tambahkan kurung kurawal pada akhir tipe, contoh seperti pada kode di atas: `map[string]int{}`.

Cara mengeset nilai pada sebuah map adalah dengan menuliskan variabel-nya, kemudian disisipkan key pada kurung siku variabel (mirip seperti cara pengaksesan elemen slice), lalu isi nilainya. Contohnya seperti chicken["februari"] = 40. Sedangkan cara pengambilan value adalah cukup dengan menyisipkan key pada kurung siku variabel.

Pengisian data pada map bersifat **overwrite**, ketika variabel sudah memiliki item dengan key yang sama, maka value lama akan ditimpa dengan value baru.

Pada pengaksesan item menggunakan key yang belum tersimpan di map, akan dikembalikan nilai default tipe data value-nya. Contohnya seperti pada kode di atas, `chicken["mei"]` menghasilkan nilai 0 (nilai

default tipe `int`), karena belum ada item yang tersimpan menggunakan key `"mei"`.

1.2.3.2 Inisialisasi Nilai Map

Zero value dari map adalah `nil`, maka tiap variabel bertipe map harus di-inisialisasi secara explisit nilai awalnya (agar tidak `nil`).

```
var data map[string]int
data["one"] = 1
// akan muncul error!

data = map[string]int{}
data["one"] = 1
// tidak ada error
```

Nilai variabel bertipe map bisa didefinisikan di awal, caranya dengan menambahkan kurung kurawal setelah tipe data, lalu menuliskan key dan value didalamnya. Cara ini sekilas mirip dengan definisi nilai array/slice namun dalam bentuk key-value.

```
// cara horizontal
var chicken1 = map[string]int{"januari": 50, "februari": 40}

// cara vertical
var chicken2 = map[string]int{
    "januari":  50,
    "februari": 40,
}
```

Key dan value dituliskan dengan pembatas tanda titik dua (`:`). Sedangkan tiap itemnya dituliskan dengan pembatas tanda koma (`,`). Khusus deklarasi dengan gaya vertikal, tanda koma perlu dituliskan setelah item terakhir.

Variabel `map` bisa diinisialisasi dengan tanpa nilai awal, caranya menggunakan tanda kurung kurawal, contoh: `map[string]int{}`. Atau bisa juga dengan menggunakan keyword `make` dan `new`. Contohnya bisa dilihat pada kode berikut. Ketiga cara di bawah ini intinya adalah sama.

```
var chicken3 = map[string]int{}
var chicken4 = make(map[string]int)
var chicken5 = *new(map[string]int)
```

Khusus inisialisasi data menggunakan keyword `new`, yang dihasilkan adalah data pointer. Untuk mengambil nilai aslinya bisa dengan menggunakan tanda asterisk (`*`). Topik pointer akan dibahas lebih detail ketika sudah masuk materi tersebut.

1.2.3.3 Iterasi Item Map Menggunakan for - range

Item variabel `map` bisa di iterasi menggunakan `for` - `range`. Cara penerapannya masih sama seperti pada slice, pembedanya data yang dikembalikan di tiap perulangan adalah key dan value, bukan indeks dan elemen. Contohnya bisa dilihat pada kode berikut.

```
var chicken = map[string]int{
    "januari":  50,
    "februari": 40,
    "maret":    34,
    "april":    67,
}

for key, val := range chicken {
    fmt.Println(key, "  \t:", val)
}
```

1.2.3.4 Menghapus Item Map

Fungsi `delete()` digunakan untuk menghapus item dengan key tertentu pada variabel map. Cara penggunaannya, dengan memasukan objek map dan key item yang ingin dihapus sebagai parameter.

```
var chicken = map[string]int{"januari": 50, "februari": 40}

fmt.Println(len(chicken)) // 2
fmt.Println(chicken)

delete(chicken, "januari")

fmt.Println(len(chicken)) // 1
fmt.Println(chicken)
```

1.2.3.5 Deteksi Keberadaan Item Dengan Key Tertentu

Ada cara untuk mengetahui apakah dalam sebuah variabel map terdapat item dengan key tertentu atau tidak, yaitu dengan memanfaatkan 2 variabel sebagai penampung nilai kembalian pengaksesan item. Return value ke-2 ini adalah opsional, isinya nilai `bool` yang menunjukkan ada atau tidaknya item yang dicari.

```
var chicken = map[string]int{"januari": 50, "februari": 40}
var value, isExist = chicken["mei"]

if isExist {
    fmt.Println(value)
} else {
    fmt.Println("item is not exists")
}
```

1.2.3.6 Kombinasi Slice & Map

Slice dan `map` bisa dikombinasikan, dan sering digunakan pada banyak kasus, contohnya seperti data array yang berisikan informasi siswa, dan banyak lainnya.

Cara menggunakannya cukup mudah, contohnya seperti `[]map[string]int`, artinya slice yang tipe tiap elemen-nya adalah `map[string]int`.

Agar lebih jelas, silakan praktekan contoh berikut.

```
var chickens = []map[string]string{
    map[string]string{"name": "chicken blue",   "gender": "male"},
    map[string]string{"name": "chicken red",    "gender": "male"},
    map[string]string{"name": "chicken yellow", "gender":
"female"},
}

for _, chicken := range chickens {
    fmt.Println(chicken["gender"], chicken["name"])
}
```

Variabel `chickens` di atas berisikan informasi bertipe `map[string]string`, yang kebetulan tiap elemen memiliki 2 key yang sama.

Jika anda menggunakan versi go terbaru, cara deklarasi slice-map bisa dipersingkat, tipe tiap elemen tidak wajib untuk dituliskan.

```
var chickens = []map[string]string{
    {"name": "chicken blue",   "gender": "male"},
    {"name": "chicken red",    "gender": "male"},
    {"name": "chicken yellow", "gender": "female"},
}
```

Dalam `[]map[string]string`, tiap elemen bisa saja memiliki key yang berbeda-beda, sebagai contoh seperti kode berikut.

```
var data = []map[string]string{
    {"name": "chicken blue", "gender": "male", "color": "brown"},
    {"address": "mangga street", "id": "k001"},
    {"community": "chicken lovers"},
}
```

1.3 Praktikum

1.3.1 Latihan Praktikum

Sebuah toko membutuhkan sistem untuk menyimpan data stok barang yang akan dijual. Data tersebut terdiri dari Kode Barang, Nama Barang, Harga, Diskon, dan Harga setelah di diskon. Buatkan sebuah program untuk mengolah data dengan ketentuan fitur berikut ini:

1. Terdapat tampilan menu untuk melakukan pengolahan data.
2. Memasukan data berupa Kode Barang, Harga, dan Diskon. Untuk harga setelah di diskon di inputkan secara otomatis oleh sistem dengan menghitung harga dengan diskon.
3. Menghapus data menurut index.
4. Menampilkan keseluruhan data ditambah dengan menampilkan jumlah data yang tersimpan.
5. Menampilkan semua barang yang di diskon > 15%
6. Menampilkan 3 barang dengan harga setelah di diskon yang paling murah.
7. Mencari barang dengan menginputkan nama barang dan menampilkan datanya.

1.3.2 Tugas Praktikum

Sebuah situs membutuhkan sebuah sistem untuk menampung dan mengolah data game terfavorit pilihan pemain. Data tersebut terdiri dari ID, Judul Game, Rating, dan Keterangan Rating. Buatkan sebuah program untuk mengolah data dengan ketentuan fitur berikut ini:

1. Terdapat tampilan menu untuk melakukan pengolahan data.
2. Memasukan Judul Game, dan Rating (0.0 – 5.0) kedalam list terurut menurut rating. Untuk Keterangan Rating dibuat langsung secara otomatis oleh sistem dengan syarat jika rating diatas 4.0 maka diisi dengan ‘good’ jika rating dari 2.0 – 4.0 diisi dengan ‘average’ dan jika rating dibawah 2.0 maka diisi dengan ‘poor’.
3. Menghapus data menurut ID.
4. Menampilkan keseluruhan data ditambah dengan menampilkan jumlah data yang tersimpan.
5. Mencari data menurut Judul Game.
6. Menampilkan top 3 game terfavorit.
7. Menampilkan seluruh data dengan rating diatas 4.0.

Note : Program tidak akan tertutup jika user tidak memilih opsi keluar

```
Contoh array kurang lebih seperti ini:

[
    [”CODM”,”good”],[”PUBG”,”average”],[”Fortnite”,”poor”]
]
```

atau

```
[
    [”CODM”,4.4],[”PUBG”,3.7],[”Fortnite”,1.9]
]
```