

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 7
STRUCT DAN ARRAY**



Disusun Oleh:

Boutefhika Nuha Ziyadatul Khair/2311102316

IF-11-05

Dosen Pengampu:

Arif Amrulloh, S. Kom., M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

A. Struct

Struct adalah tipe data komposit di Go yang memungkinkan pengelompokkan nilai-nilai terkait dari tipe berbeda.

Meskipun Go tidak mendukung kelas seperti dalam pemrograman berorientasi objek, struct berfungsi mirip dengan kelas untuk mendefinisikan tipe kompleks dengan beberapa bidang yang bisa berisi data dengan tipe yang berbeda.

Tidak seperti array atau irisan, yang homogen, struct dapat berisi data heterogen. Struct memiliki tanda tangan berikut

```
type <name> struct {  
    // field1 type  
    // field2 type  
    // field3 type  
    // ...  
}
```

Untuk mendefinisikan struct di Go, Anda menggunakan tipekata kunci diikuti dengan nama struct (di samping structkata kunci), lalu Anda memberikan daftar kolomnya yang diapit oleh kurung kurawal. Setiap kolom dideklarasikan dengan nama dan tipenya masing-masing.

```
type Person struct {  
    name string  
    age int  
    email string  
}
```

Dalam contoh di atas, kita mendefinisikan Personstruct dengan nametipe string, agetipe int, dan emailtipe string.

Go memungkinkan Anda untuk menentukan metode pada struktur, yang memungkinkan Anda untuk mengaitkan perilaku dengan data. Metode adalah fungsi yang dikaitkan dengan tipe struktur tertentu. Anda dapat menentukan metode dengan menentukan tipe penerima, yang menentukan tipe metode yang ditentukan. Pertimbangkan contoh berikut:

```
type Rectangle struct {  
    width float64  
    height float64
```

```

}

// Method defined on the Rectangle struct
func (r Rectangle) Area() float64 {
    return r.Width * r.Height
}

```

Dalam cuplikan kode di atas, kami mendefinisikan `Rectangle` struct dengan dua kolom dasar: `width` dan `height`. Kemudian, kami mendefinisikan metode yang disebut `Area()` pada `Rectangle` struct, yang menghitung dan mengembalikan luas persegi panjang. Hal ini memungkinkan instans struct `Rectangle` tidak hanya memiliki data yang terkait dengannya (misalnya, `width` dan `height`), tetapi juga perilaku (yaitu, tindakan) yang dapat dilakukan oleh instans tersebut. Dengan demikian, kami dapat menggunakan `Area()` metode:

```

rect := Rectangle{width: 8, height: 6}
area := rect.Area()
fmt.Println(area)

```

B. Array

Array dalam Go harus memiliki panjang yang tetap, artinya jika Anda menginisialisasi array agar memiliki 3 elemen, itu berarti program mengalokasikan memori sebanyak itu untuk panjang array tersebut, tidak lebih dan tidak kurang. Saat mendeklarasikan array, kita mulai dengan tanda kurung siku dengan ukuran array di dalam tanda kurung tersebut. Kemudian diikuti oleh tipe data yang dapat disimpan oleh array.

```

var siswa = [3]string

```

Jika Anda ingin menginisialisasi array dengan nilai di samping tipe data, Anda menempatkan tanda kurung kurawal di samping nilai tersebut.

```

var angka = [3]int{27, 42, 11}

```

Saat menginisialisasi array dengan nilai, Anda tidak perlu memiliki indeks array di dalam tanda kurung siku. Sebagai gantinya, Anda dapat mengisinya dengan `[...]` yang berarti

mengatur panjang array sama dengan jumlah elemen yang Anda inisialisasi.

```
var animals = [...]string{"Singa",  
"Gorila", "Kucing", "Anjing"}
```

Sebuah slice dapat diprealokasi menggunakan fungsi built-in `make`

```
// Prealokasi 10 elemen untuk sl02 dan  
sejumlah tempat tambahan  
var sl02 []int = make([]int, 10, 20)  
  
// Prealokasi 7 elemen untuk sl03 tanpa  
tempat tambahan  
var sl03 []circType = make([]circType,  
7)
```

Jika kita ingin memeriksa panjang sebuah array, Go memiliki fungsi `len()` yang akan memberikan kita panjang elemen yang kita masukkan.

```
students := [...]string{"Sean",  
"Elaina", "Clark"}  
fmt.Println(len(students))  
//metode yang digunakan untuk mencetak  
ke konsol//mengembalikan 3 karena kita  
memiliki 3 elemen dalam array
```

Fungsi built-in `append` dapat digunakan untuk menambahkan elemen ke suatu slice, dan bila perlu memperbesar tempat untuk slice tersebut.

```
/* Append elemen baru, membuat slice  
baru, dan menyimpan kembali slice baru  
ke variabel semula. Boleh juga disimpan  
ke variabel lain, sehingga variabel  
semula masih menyimpan slice yang asli.  
*/  
sl01 = append(sl01, 17)  
sl01 = append(sl01, 19, 23)
```

Map

Map adalah tipe data asosiatif yang ada di Go yang berbentuk key-value pair. Data/value yang disimpan di map selalu disertai dengan key. Key sendiri harus unik, karena digunakan sebagai penanda (atau identifier) untuk pengaksesan value yang disimpan di map.

Kalau dilihat, map mirip seperti slice, hanya saja identifier yang digunakan untuk pengaksesan bukanlah index numerik, melainkan bisa dalam tipe data apapun sesuai dengan yang diinginkan.

Cara pengaplikasian map cukup mudah, dengan menuliskan keyword map diikuti tipe data key dan value-nya. Silakan perhatikan contoh di bawah ini agar lebih jelas.

```
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
```

Item variabel map bisa di iterasi menggunakan for - range. Cara penerapannya masih sama seperti pada slice, dengan perbedaan pada map 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)
```

II. GUIDED

1. Program untuk menghitung lama parkir atau durasi waktu parkir seseorang di sebuah tempat parkir menggunakan tipe bentukan struct.

Sourcecode

```
package main  
import "fmt"  
  
type waktu struct {  
    jam, menit, detik int  
}  
  
func main() {  
    var wParkir, wPulang, durasi waktu // awal  
    var dParkir, dPulang, lParkir int // pulang  
  
    fmt.Scan(&wParkir.jam, &wParkir.menit,  
&wParkir.detik)  
    fmt.Scan(&wPulang.jam, &wPulang.menit,  
&wPulang.detik)  
    dParkir = wParkir.detik + wParkir.menit*60 +  
wParkir.jam*3600 // konversi ke detik  
    dPulang = wPulang.detik + wPulang.menit*60 +  
wPulang.jam*3600 // detik  
    lParkir = dPulang -  
dParkir // detik dari  
pulang datang  
  
    durasi.jam = lParkir / 3600  
    durasi.menit = lParkir % 3600 / 60  
    durasi.detik = lParkir % 3600 % 60 // 17  
    fmt.Printf("Lama parkir: %d jam %d menit %d detik",  
durasi.jam, durasi.menit, durasi.detik)  
}
```

Screenshoot Program

```
PS C:\Users\ASUS\Documents\Semester3\Modul7>  
go run "c:\Users\ASUS\Documents\Semester3\Modul7\Guided\guided1.go"  
7 30 0  
10 45 15  
Lama parkir: 3 jam 15 menit 15 detik  
PS C:\Users\ASUS\Documents\Semester3\Modul7>
```

Deskripsi Program

Program diatas menghitung durasi waktu parkir berdasarkan waktu datang dan waktu pulang yang diinputkan kita. Pertama, program mendeklarasikan struct **`waktu`** untuk menyimpan jam, menit, dan detik. Program kemudian menerima input waktu datang **`wParkir`** dan waktu pulang **`wPulang`**, lalu mengonversi masing-masing ke satuan detik **`dParkir`** dan **`dPulang`**. Selisih antara kedua waktu tersebut dihitung sebagai durasi total dalam detik **`IParkir`**. Selanjutnya, durasi ini dikonversi kembali ke jam, menit, dan detik supaya lebih mudah dibaca. Akhirnya, program menampilkan hasil dalam format "Lama parkir: X jam Y menit Z detik," dengan X, Y, dan Z menunjukkan jam, menit, dan detik yang dihabiskan untuk parkir.

2. Program untuk menambahkan nama-nama baru ke dalam daftar teman, tetapi hanya jika nama tersebut belum ada di dalam daftar.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengecek apakah nama sudah ada di dalam slice
func sudahAda(daftarTeman []string, nama string) bool {
    for _, teman := range daftarTeman {
        if teman == nama {
            return true
        }
    }
    return false
}

func main() {
    // Slice awal untuk daftar teman dengan beberapa data
    daftarTeman := []string{"Andi", "Budi", "Cici"}

    // Nama-nama baru yang ingin ditambahkan
    namaBaru := []string{"Dewi", "Budi", "Eka"}

    // Menambahkan nama baru hanya jika belum ada di daftar
    for _, nama := range namaBaru {
        if !sudahAda(daftarTeman, nama) {
            daftarTeman = append(daftarTeman, nama)
        } else {
```

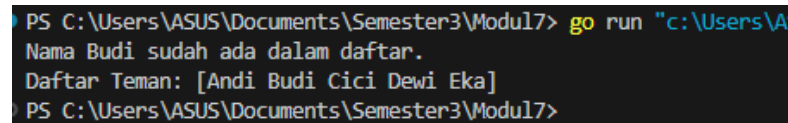
```

        fmt.Println("Nama", nama, "sudah ada dalam
daftar.")
    }
}

// Menampilkan daftar teman akhir
fmt.Println("Daftar Teman:", daftarTeman)
}

```

Screenshoot Program



```

PS C:\Users\ASUS\Documents\Semester3\Modul7> go run "c:\Users\A...
Nama Budi sudah ada dalam daftar.
Daftar Teman: [Andi Budi Cici Dewi Eka]
PS C:\Users\ASUS\Documents\Semester3\Modul7>

```

Deskripsi Program

Program diatas digunakan untuk menambahkan nama-nama baru ke dalam daftar teman tanpa duplikasi, menggunakan slice ``daftarTeman`` untuk menyimpan data awal dan ``namaBaru`` untuk nama-nama tambahan. Pertama, program mendefinisikan fungsi ``sudahAda`` yang memeriksa apakah suatu nama sudah ada di ``daftarTeman``. Lalu, program menggunakan perulangan untuk mengecek setiap nama di ``namaBaru`` dan jika nama belum ada di ``daftarTeman``, maka nama tersebut ditambahkan, sedangkan jika sudah ada, program menampilkan pesan bahwa nama tersebut sudah ada. Dan program akan menampilkan daftar teman yang telah diperbarui, serta menampilkan hasil akhir tanpa duplikasi.

3. Program menyimpan dan menampilkan daftar harga beberapa buah yang menggunakan struktur data map.

Sourcecode

```

package main

import (
    "fmt"
)

func main() {
    // Membuat map dengan nama buah sebagai kunci dan
    harga sebagai nilai
    hargaBuah := map[string]int{
        "Apel": 5000,
        "Pisang": 3000,
        "Mangga": 7000,
    }

    // Menampilkan harga dari setiap buah

```



```

    fmt.Println("Harga Buah:")
    for buah, harga := range hargaBuah {
        fmt.Printf("%s: Rp%d\n", buah, harga)
    }

    fmt.Print("Harga      buah      Mangga      =      ",
hargaBuah["Mangga"])
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3\Modul7>
go run "c:\Users\ASUS\Documents\Semester3\Modul7\Guided\guided3.go"
Harga Buah:
Apel: Rp5000
Pisang: Rp3000
Mangga: Rp7000
Harga buah Mangga = 7000
PS C:\Users\ASUS\Documents\Semester3\Modul7>

```

Deskripsi Program

Program diatas digunakan untuk menyimpan dan menampilkan harga beberapa buah menggunakan struktur ``map`` bernama ``hargaBuah`` yang digunakan dengan nama buah sebagai kunci dan harga sebagai nilai. Pertama, program mendeklarasikan ``hargaBuah`` dengan beberapa pasangan kunci-nilai: "Apel" berharga 5000, "Pisang" 3000, dan "Mangga" 7000. Program kemudian menampilkan daftar harga buah menggunakan perulangan ``for range`` untuk mengakses dan mencetak setiap buah beserta harganya. Terakhir, program secara khusus menampilkan harga untuk buah "Mangga" dengan mengakses nilainya langsung dari ``map`` menggunakan kunci "Mangga".

III. UNGUIDED

1. Suatu lingkaran didefinisikan dengan koordinat titik pusat (cx , cy) dengan radius r . Apabila diberikan dua buah lingkaran, maka tentukan posisi sebuah titik sembarang (x , y) berdasarkan dua lingkaran tersebut. Gunakan tipe bentukan titik untuk menyimpan koordinat, dan tipe bentukan lingkaran untuk menyimpan titik pusat lingkaran dan radiusnya.
Masukan terdiri dari beberapa tiga baris. Baris pertama dan kedua adalah koordinat titik pusat dan radius dari lingkaran 1 dan lingkaran 2, sedangkan baris ketiga adalah koordinat titik sembarang. Asumsi sumbu x dan y dari semua titik dan juga radius direpresentasikan dengan bilangan bulat.

Keluaran berupa string yang menyatakan posisi titik "Titik di dalam lingkaran 1 dan 2", "Titik di dalam lingkaran 1", "Titik di dalam lingkaran 2", atau "Titik di luar lingkaran 1 dan 2".

Fungsi untuk menghitung jarak titik (a, b) dan (c, d) dimana rumus jarak adalah:

$$jarak = \sqrt{(a - c)^2 + (b - d)^2}$$

dan juga fungsi untuk menentukan posisi sebuah titik sembarang berada di dalam suatu lingkaran atau tidak

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

// Definisi tipe untuk titik dan lingkaran
type Titik struct {
    x, y int
}

type Lingkaran struct {
    titikPusat Titik
    radius      int
}

// Fungsi untuk menghitung jarak antara dua titik
func jarak(p, q Titik) float64 {
    return math.Sqrt(float64((q.x-p.x)*(q.x-p.x) +
(q.y-p.y)*(q.y-p.y)))
}

// Fungsi untuk menentukan apakah titik berada di
dalam lingkaran
func didalam(c Lingkaran, p Titik) bool {
    return jarak(c.titikPusat, p) <=
float64(c.radius)
}

// Fungsi untuk mengecek posisi titik terhadap dua
lingkaran
func cekPosisi(p Titik, l1, l2 Lingkaran) string {
    var dalamLingkaran1, dalamLingkaran2 bool
    dalamLingkaran1 = didalam(l1, p)
    dalamLingkaran2 = didalam(l2, p)

    if dalamLingkaran1 && dalamLingkaran2 {
```

```

        return "Titik di dalam lingkaran 1 dan 2"
    } else if dalamLingkaran1 {
        return "Titik di dalam lingkaran 1"
    } else if dalamLingkaran2 {
        return "Titik di dalam lingkaran 2"
    } else {
        return "Titik di luar lingkaran 1 dan 2"
    }
}

func main() {
    // Deklarasi variabel untuk lingkaran 1
    var cx1, cy1, r1 int
    fmt.Scan(&cx1, &cy1, &r1)
    var lingkaran1 Lingkaran
    lingkaran1.titikPusat = Titik{cx1, cy1}
    lingkaran1.radius = r1

    // Deklarasi variabel untuk lingkaran 2
    var cx2, cy2, r2 int
    fmt.Scan(&cx2, &cy2, &r2)
    var lingkaran2 Lingkaran
    lingkaran2.titikPusat = Titik{cx2, cy2}
    lingkaran2.radius = r2

    // Deklarasi variabel untuk titik sembarang
    var x, y int
    fmt.Scan(&x, &y)
    var titik Titik
    titik.x = x
    titik.y = y

    // Tentukan posisi titik
    var hasil string
    hasil = cekPosisi(titik, lingkaran1, lingkaran2)
    fmt.Println(hasil)
}

```

Screenshoot Output

```
PS C:\Users\ASUS\Documents\Semester3\Modul7>
● go run "c:\Users\ASUS\Documents\Semester3\Modul7\Unguided\unguided1.go"
1 1 5
8 8 4
2 2
Titik di dalam lingkaran 1
PS C:\Users\ASUS\Documents\Semester3\Modul7> go run "c:\Users
1 2 3
● 4 5 6
7 8
Titik di dalam lingkaran 2
PS C:\Users\ASUS\Documents\Semester3\Modul7> go run "c:\Users\ASUS\Docume
5 10 15
-15 4 20
0 0
Titik di dalam lingkaran 1 dan 2
PS C:\Users\ASUS\Documents\Semester3\Modul7> go run "c:\Users\ASUS\Docume
1 1 5
8 8 4
15 20
Titik di luar lingkaran 1 dan 2
PS C:\Users\ASUS\Documents\Semester3\Modul7> []
```

Deskripsi Program

Program di atas mengecek posisi sebuah titik terhadap dua lingkaran yang diberikan. Program mendeklarasikan dua struktur data, yaitu **Titik** untuk menyimpan koordinat titik dan **Lingkaran** untuk menyimpan informasi pusat dan radius lingkaran. Pertama program akan menerima input koordinat pusat dan radius untuk dua lingkaran, dan koordinat titik yang akan diperiksa. Fungsi **jarak** digunakan untuk menghitung jarak antara titik pusat lingkaran dan titik yang diperiksa. Fungsi **didalam** kemudian memeriksa apakah titik berada di dalam lingkaran dengan membandingkan jarak titik ke pusat lingkaran dengan radius lingkaran. Fungsi **cekPosisi** memeriksa apakah titik berada di dalam salah satu atau kedua lingkaran, atau di luar keduanya. Berdasarkan hasil pemeriksaan tadi, lalu program akan mencetak salah satu dari empat kemungkinan hasil: titik berada di dalam kedua lingkaran, hanya lingkaran pertama, hanya lingkaran kedua, atau di luar kedua lingkaran.

2. Sebuah array digunakan untuk menampung sekumpulan bilangan bulat. Buatlah program yang digunakan untuk mengisi array tersebut sebanyak N elemen nilai. Asumsikan array memiliki kapasitas penyimpanan data sejumlah elemen tertentu. Program dapat menampilkan beberapa informasi berikut:
 - a. Menampilkan keseluruhan isi dari array.
 - b. Menampilkan elemen-elemen array dengan indeks ganjil saja.

- c. Menampilkan elemen-elemen array dengan indeks genap saja (asumsi indeks ke-0 adalah genap).
- d. Menampilkan elemen-elemen array dengan indeks kelipatan bilangan x. x bisa diperoleh dari masukan pengguna.
- e. Menghapus elemen array pada indeks tertentu, asumsi indeks yang hapus selalu valid.
- f. Tampilkan keseluruhan isi dari arraynya, pastikan data yang dihapus tidak tampil
- g. Menampilkan rata-rata dari bilangan yang ada di dalam array.
- h. Menampilkan standar deviasi atau simpangan baku dari bilangan yang ada di dalam array tersebut.
- i. Menampilkan frekuensi dari suatu bilangan tertentu di dalam array yang telah diisi tersebut.

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

// a. Tampilkan seluruh isi array
func tampilkanArray(arr []int) {
    fmt.Println("Isi array:", arr)
}

// b. Tampilkan elemen dengan indeks ganjil
func tampilkanIndeksGanjil(arr []int) {
    fmt.Print("Elemen dengan indeks ganjil: ")
    for i := 1; i < len(arr); i += 2 {
        fmt.Print(arr[i], " ")
    }
    fmt.Println()
}

// c. Tampilkan elemen dengan indeks genap
func tampilkanIndeksGenap(arr []int) {
    fmt.Print("Elemen dengan indeks genap: ")
    for i := 0; i < len(arr); i += 2 {
        fmt.Print(arr[i], " ")
    }
    fmt.Println()
}

// d. Tampilkan elemen dengan indeks kelipatan x
func tampilkanKelipatanX(arr []int, x int) {
```

```

        fmt.Printf("Elemen dengan indeks kelipatan %d: ",
x)
        for i := 0; i < len(arr); i++ {
            if i%x == 0 {
                fmt.Print(arr[i], " ")
            }
        }
        fmt.Println()
    }

// e. Hapus elemen pada indeks tertentu
func hapusElemen(arr []int, indeks int) []int {
    return append(arr[:indeks], arr[indeks+1:]...)
}

// f. Hitung rata-rata elemen dalam array
func hitungRataRata(arr []int) float64 {
    if len(arr) == 0 {
        return 0
    }
    jumlah := 0
    for _, nilai := range arr {
        jumlah += nilai
    }
    return float64(jumlah) / float64(len(arr))
}

// g. Hitung standar deviasi elemen dalam array
func hitungStandarDeviasi(arr []int) float64 {
    if len(arr) == 0 {
        return 0
    }
    rataRata := hitungRataRata(arr)
    var jumlah float64
    for _, nilai := range arr {
        jumlah += math.Pow(float64(nilai)-rataRata, 2)
    }
    return math.Sqrt(jumlah / float64(len(arr)))
}

// h. Hitung frekuensi dari suatu bilangan tertentu
dalam array
func hitungFrekuensi(arr []int, bilangan int) int {
    jumlah := 0
    for _, nilai := range arr {
        if nilai == bilangan {
            jumlah++
        }
    }
}

```

```

        return jumlah
    }

func main() {
    var n int
    fmt.Print("Masukkan jumlah elemen array: ")
    fmt.Scan(&n)

    arr := make([]int, n)
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan elemen ke-%d: ", i+1)
        fmt.Scan(&arr[i])
    }

    tampilkanArray(arr)
    tampilkanIndeksGanjil(arr)
    tampilkanIndeksGenap(arr)

    var x int
    fmt.Print("Masukkan bilangan x untuk menampilkan
elemen dengan indeks kelipatan x: ")
    fmt.Scan(&x)
    tampilkanKelipatanX(arr, x)

    var indeksUntukDihapus int
    fmt.Print("Masukkan indeks elemen yang ingin
dihapus: ")
    fmt.Scan(&indeksUntukDihapus)
    arr = hapusElemen(arr, indeksUntukDihapus)
    fmt.Println("Array setelah penghapusan elemen pada
indeks", indeksUntukDihapus, ":", arr)

    fmt.Printf("Rata-rata elemen dalam array: %.2f\n",
hitungRataRata(arr))
    fmt.Printf("Standar deviasi elemen dalam array:
%.2f\n", hitungStandarDeviasi(arr))

    var cari int
    fmt.Print("Masukkan bilangan untuk menghitung
frekuensi: ")
    fmt.Scan(&cari)
    freq := hitungFrekuensi(arr, cari)
    fmt.Printf("Frekuensi bilangan %d dalam array:
%d\n", cari, freq)
}

```

Screenshoot Program

```
PS C:\Users\ASUS\Documents\Semester3\Modul7>
go run "c:\Users\ASUS\Documents\Semester3\Modul7\Unguided\unguided2.go"
Masukkan jumlah elemen array: 5
Masukkan elemen ke-1: 10
Masukkan elemen ke-2: 20
Masukkan elemen ke-3: 40
Masukkan elemen ke-4: 60
Masukkan elemen ke-5: 30
Isi array: [10 20 40 60 30]
Elemen dengan indeks ganjil: 20 60
Elemen dengan indeks genap: 10 40 30
Masukkan bilangan x untuk menampilkan elemen dengan indeks kelipatan x: 2
Elemen dengan indeks kelipatan 2: 10 40 30
Masukkan indeks elemen yang ingin dihapus: 1
Array setelah penghapusan elemen pada indeks 1 : [10 40 60 30]
Rata-rata elemen dalam array: 35.00
Standar deviasi elemen dalam array: 18.03
Masukkan bilangan untuk menghitung frekuensi: 20
Frekuensi bilangan 20 dalam array: 0
PS C:\Users\ASUS\Documents\Semester3\Modul7>
```

Deskripsi Program

Program diatas menginput jumlah elemen dan nilai-nilai elemen array dari pengguna, kemudian melakukan beberapa operasi berdasarkan fungsi yang telah didefinisikan. Fungsi pertama menampilkan seluruh isi array, fungsi untuk menampilkan elemen-elemen pada indeks ganjil dan genap. Program kemudian meminta input bilangan x untuk menampilkan elemen-elemen yang berada pada indeks kelipatan x. Selanjutnya, pengguna diminta untuk menginputkan indeks elemen yang ingin dihapus dari array, yang kemudian diperbarui dan ditampilkan. Program juga menghitung rata-rata dan standar deviasi elemen dalam array, serta menghitung frekuensi kemunculan suatu bilangan tertentu dalam array.

3. Sebuah program digunakan untuk menyimpan dan menampilkan nama-nama klub yang memenangkan pertandingan bola pada suatu grup pertandingan. Buatlah program yang digunakan untuk merekap skor pertandingan bola 2 buah klub bola yang berlaga.

Pertama-tama program meminta masukan nama-nama klub yang bertanding, kemudian program meminta masukan skor hasil pertandingan kedua klub tersebut. Yang disimpan dalam array adalah nama-nama klub yang menang saja.

Proses input skor berhenti ketika skor salah satu atau kedua klub tidak valid (negatif). Di akhir program, tampilkan daftar klub yang memenangkan pertandingan.

Sourcecode

```
package main

import "fmt"
```



```

func main() {
    // Variabel untuk nama klub
    var klubA, klubB string

    // Meminta input nama klub A dan B
    fmt.Print("Klub A: ")
    fmt.Scan(&klubA)
    fmt.Print("Klub B: ")
    fmt.Scan(&klubB)

    // Array untuk menyimpan hasil pertandingan
    var hasil []string

    // Variabel untuk skor pertandingan
    var skorA, skorB int

    // Input dan proses pertandingan
    for i := 1; ; i++ {
        // Meminta input skor pertandingan
        fmt.Printf("Pertandingan %d:\n", i)
        fmt.Printf("%s : ", klubA)
        fmt.Scan(&skorA)
        fmt.Printf("%s : ", klubB)
        fmt.Scan(&skorB)

        // Memeriksa apakah ada skor yang negatif
        if skorA < 0 || skorB < 0 {
            break
        }

        // Menyimpan pemenang berdasarkan skor
        if skorA > skorB {
            hasil = append(hasil, klubA)
        } else if skorA < skorB {
            hasil = append(hasil, klubB)
        } else {
            hasil = append(hasil, "Draw")
        }
    }

    // Menampilkan hasil pertandingan
    for i := 0; i < len(hasil); i++ {
        fmt.Printf("Hasil %d : %s\n", i+1, hasil[i])
    }
    fmt.Println("Pertandingan selesai")
}

```

Screenshoot Program

```
PS C:\Users\ASUS\Documents\Semester3\Modul7>
go run "c:\Users\ASUS\Documents\Semester3\Modul7\Ung
ded3.go"
● Klub A: MU
Klub B: Inter
Pertandingan 1:
MU : 2
Inter : 0
Pertandingan 2:
MU : 1
Inter : 2
Pertandingan 3:
MU : 2
Inter : 2
Pertandingan 4:
MU : 0
Inter : 1
Pertandingan 5:
MU : 3
Inter : 2
Pertandingan 6:
MU : 1
Inter : 0
Pertandingan 7:
MU : 5
Inter : 2
Pertandingan 8:
MU : 2
Inter : 3
Pertandingan 9:
MU : -1
Inter : 2
Hasil 1 : MU
Hasil 2 : Inter
Hasil 3 : Draw
Hasil 4 : Inter
Hasil 5 : MU
Hasil 6 : MU
Hasil 7 : MU
Hasil 8 : Inter
Pertandingan selesai
○ PS C:\Users\ASUS\Documents\Semester3\Modul7> □
```

Deskripsi Program

Program diatas menginput nama dua klub sepak bola (klub A dan klub B) dari pengguna, lalu menjalankan sebuah loop untuk mencatat hasil pertandingan antara kedua klub tersebut. Di dalam loop, program diminta menginput skor pertandingan dan memeriksa apakah ada skor negatif, yang akan menghentikan input. Jika skor valid, program membandingkan skor kedua klub dan menyimpan hasil pertandingan, apakah klub A menang, klub B menang, atau hasilnya seri. Setelah

pertandingan selesai, program menampilkan hasil dari setiap pertandingan yang telah dimasukkan, diakhiri dengan pesan "Pertandingan selesai". Program terus meminta input pertandingan hingga skor negatif dimasukkan.

4. Sebuah array digunakan untuk menampung sekumpulan karakter, Anda diminta untuk membuat sebuah subprogram untuk melakukan membalikkan urutan isi array dan memeriksa apakah membentuk palindrom.

Sourcecode

```
package main

import "fmt"

const NMAX int = 127 // NMAX adalah konstanta yang
mendefinisikan ukuran maksimal array

type tabel [NMAX]rune // Tabel adalah tipe data array
yang berisi elemen bertipe rune (karakter Unicode)

var tab tabel // Variabel global tab bertipe tabel untuk
menyimpan teks yang dimasukkan
var m int // Variabel m untuk menyimpan jumlah
karakter yang dimasukkan

// Prosedur isiArray untuk memasukkan karakter-karakter
ke dalam array
func isiArray(t *tabel, n *int) {
    fmt.Print("Teks: ")
    var input rune
    *n = 0 // menginisialisasi panjang array dengan 0
    for *n < NMAX {
        fmt.Scanf("%c", &input)
        if input == '.' { // Jika karakter yang
            dimasukkan adalah '.', keluar dari loop
            break
        }
        t[*n] = input
        (*n)++
    }
}

// Prosedur cetakArray untuk mencetak isi array
func cetakArray(t tabel, n int) {
    fmt.Print("Reverse Teks: ")
    for i := 0; i < n; i++ {
```

```

        fmt.Printf("%c", t[i]) // Menampilkan karakter
        satu per satu
    }
    fmt.Println()
}

// Prosedur balikanArray untuk membalikkan urutan elemen
dalam array
func balikanArray(t *tabel, n int) {
    for i := 0; i < n/2; i++ {
        t[i], t[n-i-1] = t[n-i-1], t[i] // Menukar elemen
        yang berada di posisi i dan n-i-1
    }
}

func main() {
    var m int
    isiArray(&tab, &m) // isi array tab dengan
    memanggil prosedur isiArray
    balikanArray(&tab, m) // balikkan isi array tab
    dengan memanggil balikanArray
    cetakArray(tab, m) // cetak isi array tab
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3\Modul7>
go run "c:\Users\ASUS\Documents\Semester3\Modul7\Unguided\unguided4.go"
Teks: SENANG.
Reverse Teks: GNANES
PS C:\Users\ASUS\Documents\Semester3\Modul7>

```

Deskripsi Program

Program diatas meminta pengguna menginputkan teks karakter ke dalam sebuah array, kemudian membalikkan urutan karakter dan menampilkannya dalam urutan terbalik. Program dimulai dengan mendeklarasikan konstanta ``NMAX`` sebagai ukuran maksimal array dan mendefinisikan tipe ``tabel`` sebagai array yang menyimpan karakter ``rune``. Fungsi ``isiArray`` digunakan untuk meminta input karakter dari pengguna dan menyimpannya dalam array hingga pengguna memasukkan titik (``.``), yang menandakan akhir input. Fungsi ``balikanArray`` membalikkan urutan elemen dalam array dengan menukar posisi elemen dari depan dan belakang. Setelah array terbalik, fungsi ``cetakArray`` digunakan untuk mencetak isi array dalam urutan terbalik. Program menggabungkan ketiga prosedur untuk menghasilkan output teks yang dibalik dari input yang diberikan.

Modifikasi program tersebut dengan menambahkan fungsi palindrom. Tambahkan instruksi untuk memanggil fungsi tersebut dan menampilkan hasilnya pada program utama.

Sourcecode

```
package main

import "fmt"

const NMAX int = 127 // NMAX adalah konstanta yang
mendefinisikan ukuran maksimal array

type tabel [NMAX]rune // Tabel adalah tipe data array
yang berisi elemen bertipe rune (karakter Unicode)

var tab tabel // Variabel global tab bertipe tabel untuk
menyimpan teks yang dimasukkan
var m int // Variabel m untuk menyimpan jumlah
karakter yang dimasukkan

// Prosedur isiArray untuk memasukkan karakter-karakter
ke dalam array
func isiArray(t *tabel, n *int) {
    fmt.Print("Teks      : ")
    var input string // Menggunakan string untuk membaca
    satu baris teks
    fmt.Scanln(&input)
    *n = len(input) // Menyimpan panjang teks yang
    dimasukkan
    for i := 0; i < *n; i++ {
        t[i] = rune(input[i]) // Menyimpan setiap
        karakter dalam array
    }
}

// Prosedur balikanArray untuk membalikkan urutan elemen
dalam array
func balikanArray(t *tabel, n int) {
    for i := 0; i < n/2; i++ {
        t[i], t[n-i-1] = t[n-i-1], t[i] // Menukar elemen
        yang berada di posisi i dan n-i-1
    }
}

// Membalikkan array dan memeriksa apakah array asli dan
dibalik sama
func palindrom(t tabel, n int) bool {
    var temp tabel
```

```

        copy(temp[:], t[:n])    // Menyalin array asli ke
temp
        balikArray(&temp, n) // Membalikkan array temp
        for i := 0; i < n; i++ {
            if t[i] != temp[i] {
                return false // Tidak palindrom
            }
        }
        return true // Palindrom
    }

func main() {
    isiArray(&tab, &m)          // isi array tab dengan
memanggil prosedur isiArray
    if palindrom(tab, m) { // Memeriksa apakah teks
adalah palindrom
        fmt.Println("Palindrom ? True")
    } else {
        fmt.Println("Palindrom ? False")
    }
}

```

Screenshoot Program

```

PS C:\Users\ASUS\Documents\Semester3\Modul7>
go run "c:\Users\ASUS\Documents\Semester3\Modul7\Unguided\unguided4b.go"
Teks      : SENANG
Palindrom ? False
PS C:\Users\ASUS\Documents\Semester3\Modul7> go run "c:\Users\ASUS\Documents
\Semester3\Modul7\Unguided\unguided4b.go"
Teks      : KATAK
Palindrom ? True
PS C:\Users\ASUS\Documents\Semester3\Modul7>

```

Deskripsi Program

Program diatas meminta pengguna untuk memasukkan sebuah teks, kemudian memeriksa apakah teks tersebut merupakan palindrom, apakah teks tersebut tetap sama jika dibaca dari belakang. Pertama, program membaca input berupa teks dan menyimpannya dalam array ``tab``. Fungsi ``isiArray`` digunakan untuk menyimpan karakter-karakter dari teks ke dalam array ``tab``. Fungsi ``balikanArray`` membalikkan urutan elemen dalam array. Setelah itu, fungsi ``palindrom`` digunakan untuk memeriksa apakah teks asli dan teks yang telah dibalik sama, dengan menyalin array asli ke array sementara dan membalikkan array sementara. Jika kedua array sama, maka teks tersebut adalah palindrom dan program mencetak **"Palindrom ? True"**, jika tidak, program mencetak **"Palindrom ? False"**.

IV. KESIMPULAN

Seperti yang kita ketahui ada berbagai tipe data dan struktur dalam bahasa pemrograman Go (Golang), seperti struct, array, slice, dan map, sangat berguna untuk mempermudah pengelolaan data dalam berbagai masalah. Struct efektif untuk merepresentasikan tipe data kompleks, array cocok untuk data dengan jumlah tetap, sementara slice lebih fleksibel dalam penanganan ukuran data yang dinamis. Map memungkinkan pengelolaan data berbasis pasangan kunci-nilai yang memudahkan pencarian dan pengelompokan. Ini diimplementasikan melalui berbagai studi kasus, seperti perhitungan durasi waktu, pengelolaan daftar nama, dan pemeriksaan posisi titik terhadap lingkaran, menunjukkan efisiensi dan fleksibilitasnya dalam menyelesaikan masalah. Selain itu, penggunaan algoritma tambahan untuk analisis data, seperti menghitung rata-rata dan standar deviasi, menegaskan kemampuan Go dalam menangani berbagai tugas analitis dan logis dengan struktur yang sederhana dan efisien.

V. REFERENSI

- [1] Udacity. (2023, June). *Go (Golang) Structs*. Diakses dari <https://www.udacity.com/blog/2023/06/go-golang-structs.html>
- [2] Sdever. (t.t.). **Working with Arrays in Go (Golang)**. Diakses dari https://sdever.medium.com/working-with-arrays-in-go-golang-9b6738b8995c(<https://sdever.medium.com/working-with-arrays-in-go-golang-9b6738b8995c>).
- [3] Agung, N. (t.t.). *Map pada Golang*. Diakses dari <https://dasarpemrogramangolang.novalagung.com/A-map.html>.