

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL VII
STRUCK DAN ARRAY**



**Disusun Oleh :
SYAHRUL ROMADHONI / 2311102261
S1 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

Definisi Struct dan Array dalam Golang

Memahami struktur data adalah langkah pertama untuk menjadi ahli dalam pemrograman Go. Struktur data membantu menyimpan dan mengorganisir data dalam program, memungkinkan kita untuk melakukan operasi data secara efektif. Misalnya, array dalam Go digunakan untuk menyimpan kumpulan elemen tipe data yang sama. Hal ini memudahkan dalam pengaksesan dan pengelolaan data.

Di Go, struktur data tidak hanya terbatas pada array, tetapi juga mencakup tipe data lain seperti slice, map, dan struct. Slice adalah versi dinamis dari array yang lebih fleksibel dalam penggunaan karena ukurannya yang bisa berubah. Map, di sisi lain, menyimpan data dalam pasangan kunci-nilai, sangat berguna untuk pencarian cepat dan efisien.

Struct dalam Go memungkinkan kita untuk mendefinisikan tipe data yang lebih kompleks, menggabungkan berbagai item data dalam satu unit. Ini sering digunakan untuk merepresentasikan objek dengan atribut yang berbeda-beda. Misalnya, kita bisa mendefinisikan struct `Mahasiswa` dengan atribut `nama`, `umur`, dan `jurusan`.

Mengenal Map dan Struct

Map di Go adalah tipe data yang menyimpan pasangan kunci-nilai, sering digunakan untuk mencari data secara efisien. Kunci dalam map harus unik dan memiliki tipe yang sama, sementara nilai bisa berupa tipe apa pun. Ini membuat map sangat fleksibel dan berguna untuk struktur data seperti kamus atau database sederhana.

```
mahasiswa := map[string]int{
    "Alice": 23,
    "Bob": 25,
}
```

Sementara itu, struct di Go memungkinkan kita untuk menggabungkan beberapa tipe data yang berbeda dalam satu entitas yang lebih besar. Ini sangat bermanfaat dalam merepresentasikan objek dengan atribut yang beragam, mirip dengan kelas dalam pemrograman berorientasi objek.

Struct mendukung definisi metode, yang memungkinkan kita untuk menambahkan perilaku kepada data.

```
type          Buku          struct      {  
    Judul      string  
    Penulis    string  
    Tahun      int  
}
```

```
buku := Buku{"Go Programming", "John Doe", 2020}
```

Memahami cara kerja map dan struct esensial untuk mengelola data yang kompleks dalam pemrograman Go. Keduanya menyediakan struktur yang kuat untuk mengatur data dan mendukung konsep pemrograman yang lebih tinggi.

II. GUIDED

Soal Studi Case

1.

SOURCECODE

```
package main

import "fmt"

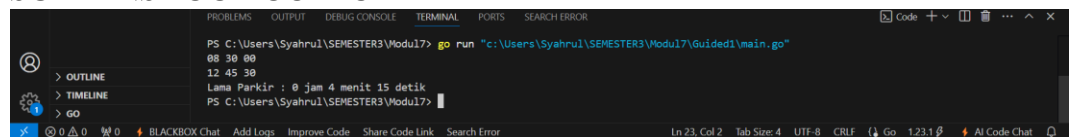
type waktu struct {
    jam, menit, detik int
}

func main() {
    var wParkir, wPulang, durasi waktu
    var dParkir, dPulang, lParkir int

    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 OUTPUT



DESKRIPSI PROGRAM

Program di atas adalah aplikasi Go yang menghitung durasi waktu parkir berdasarkan waktu kedatangan dan waktu pulang yang diinput oleh pengguna. Program menggunakan **struct** bernama waktu untuk menyimpan nilai jam, menit, dan detik. Waktu kedatangan dan pulang dikonversi ke dalam satuan detik untuk memudahkan perhitungan selisih waktu. Selisih ini dihitung dalam detik, kemudian dikonversi kembali menjadi jam, menit, dan detik untuk menampilkan hasilnya dalam format yang lebih mudah dibaca. Program ini menunjukkan cara penggunaan **struct** untuk

menyederhanakan pengelolaan data waktu dan perhitungan durasi dengan logika matematika sederhana.

2.

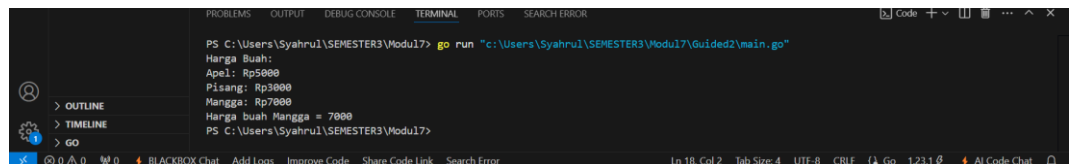
SOURCECODE

```
package main
import (
    "fmt"
)

func main() {
    hargaBuah := map[string]int{
        "Apel": 5000,
        "Pisang": 3000,
        "Mangga": 7000,
    }
    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 OUTPUT



DESKRIPSI PROGRAM

Program di atas adalah aplikasi Go sederhana yang menggunakan **map** untuk menyimpan pasangan data berupa nama buah sebagai kunci dan harga buah sebagai nilai. Program mendefinisikan daftar harga buah (apel, pisang, dan mangga) dalam map hargaBuah dan mencetak semua pasangan data menggunakan perulangan for range. Selain itu, program juga menampilkan harga spesifik buah mangga dengan mengakses nilai map menggunakan kunci "Mangga". Map digunakan karena memungkinkan akses cepat ke nilai berdasarkan kunci, menjadikannya ideal untuk menyimpan data asosiatif seperti daftar harga.

3.

SOURCECODE

```
package main
```

```

import (
    "fmt"
)

func sudahAda(daftarTeman []string, nama string) bool {
    for _, teman := range daftarTeman {
        if teman == nama {
            return true
        }
    }
    return false
}

func main() {

    daftarTeman := []string{"Andi", "Budi", "Cici"}

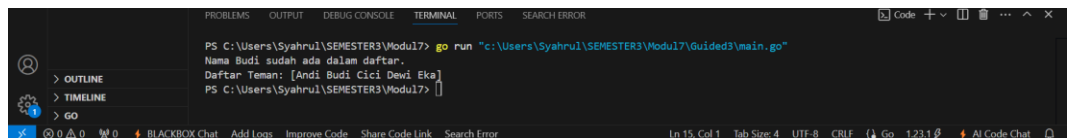
    namaBaru := []string{"Dewi", "Budi", "Eka"}

    for _, nama := range namaBaru {
        if !sudahAda(daftarTeman, nama) {
            daftarTeman = append(daftarTeman, nama)
        } else {
            fmt.Println("Nama", nama, "sudah ada dalam daftar.")
        }
    }

    fmt.Println("Daftar Teman:", daftarTeman)
}

```

SCREENSHOOT OUTPUT



```

PS C:\Users\Syahrul\SEMESTER3\Modul7> go run "c:\Users\Syahrul\SEMESTER3\Modul7\Guided3\main.go"
Nama Budi sudah ada dalam daftar.
Daftar Teman: [Andi Budi Cici Dewi Eka]
PS C:\Users\Syahrul\SEMESTER3\Modul7>

```

DESKRIPSI PROGRAM

Program di atas adalah aplikasi sederhana yang memperbarui daftar teman dengan menambahkan nama baru, jika nama tersebut belum ada dalam daftar. Program dimulai dengan daftar teman awal (daftarTeman) dan daftar nama baru (namaBaru). Untuk setiap nama dalam namaBaru, program memeriksa apakah nama tersebut sudah ada di daftarTeman menggunakan fungsi sudahAda, yang melakukan pencarian linier dalam array. Jika nama belum ada, nama tersebut ditambahkan ke daftar; jika sudah ada, program mencetak pesan bahwa nama tersebut telah ada. Di akhir eksekusi, program

menampilkan daftar teman yang telah diperbarui. Program ini menunjukkan cara sederhana untuk mengelola data array dengan mencegah duplikasi.

III. UNGUIDED

Soal Studi Case

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".

Contoh

No	Masukan	Keluaran
1	1 1 5 8 8 4 2 2	Titik di dalam lingkaran 1
2	1 2 3 4 5 6 7 8	Titik di dalam lingkaran 2
3	5 10 15 -15 4 20 0 0	Titik di dalam lingkaran 1 dan 2
4	1 1 5 8 8 4 15 20	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.


```

function jarak(p, q : titik) -> real
{Mengembalikan jarak antara titik p(x,y) dan titik q(x,y)}

function didalam(c:lingkaran, p:titik) -> boolean
{Mengembalikan true apabila titik p(x,y) berada di dalam lingkaran c yang
memiliki titik pusat (cx,cy) dan radius r}

```

Catatan: Lihat paket **math** dalam lampiran untuk menggunakan fungsi **math.Sqrt()** untuk menghitung akar kuadrat.

Sourcecode

```

package main

import (
    "fmt"
    "math"
)

type Titik struct {
    x, y int
}

type Lingkaran struct {
    cx, cy, r int
}

func jarak(p1, p2 Titik) float64 {
    return math.Sqrt(float64((p1.x-p2.x)*(p1.x-p2.x) + (p1.y-p2.y)*(p1.y-
p2.y)))
}

func diDalam(c Lingkaran, p Titik) bool {
    jarakTitik := jarak(Titik{c.cx, c.cy}, p)
    return jarakTitik <= float64(c.r)
}

func main() {
    lingkaran1 := Lingkaran{cx: 2, cy: 1, r: 5}
    lingkaran2 := Lingkaran{cx: 8, cy: 4, r: 2}
    titik := Titik{x: 3, y: 2}

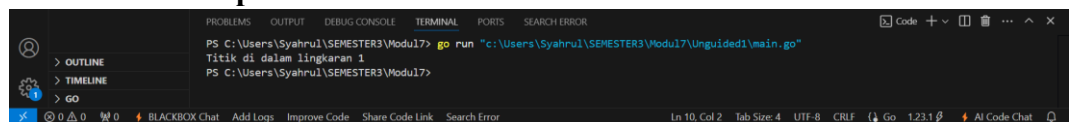
    dalamL1 := diDalam(lingkaran1, titik)
    dalamL2 := diDalam(lingkaran2, titik)

    // Output hasil
    if dalamL1 && dalamL2 {
        fmt.Println("Titik di dalam lingkaran 1 dan 2")
    } else if dalamL1 {

```

```
        fmt.Println("Titik di dalam lingkaran 1")
    } else if dalamL2 {
        fmt.Println("Titik di dalam lingkaran 2")
    } else {
        fmt.Println("Titik di luar lingkaran 1 dan 2")
    }
}
```

Screenshoot Output



Deskripsi Program

Program ini digunakan untuk menentukan posisi sebuah titik terhadap dua lingkaran yang memiliki koordinat pusat dan radius tertentu. Program memanfaatkan dua struktur data (struct), yaitu **Titik** untuk merepresentasikan koordinat (x, y) dan **Lingkaran** untuk menyimpan pusat lingkaran (cx, cy) dan jari-jari (r). Fungsi jarak menghitung jarak Euclidean antara dua titik, sedangkan fungsi `diDalam` memeriksa apakah sebuah titik berada di dalam lingkaran dengan membandingkan jaraknya ke pusat lingkaran dengan radius lingkaran. Di dalam fungsi `main`, program memeriksa apakah titik tertentu berada di dalam salah satu, kedua, atau di luar lingkaran, dan mencetak hasilnya ke layar.

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. Tampilkan keseluruhan isi dari arraynya, pastikan data yang dihapus tidak tampil
- f. Menampilkan rata-rata dari bilangan yang ada di dalam array.
- g. Menampilkan standar deviasi atau simpangan baku dari bilangan yang ada di dalam array tersebut.
- h. Menampilkan frekuensi dari suatu bilangan tertentu di dalam array yang telah diisi tersebut.

Sourcecode

```
package main

import (
    "fmt"
    "math"
)

func tampilkanArray(arr []int) {
    fmt.Println("Isi array:", arr)
}

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

func tampilkanIndeksGenap(arr []int) {
    fmt.Print("Elemen dengan indeks genap: ")
```

```

    for i := 0; i < len(arr); i += 2 {
        fmt.Print(arr[i], " ")
    }
    fmt.Println()
}

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

func hapusElemen(arr []int, idx int) []int {
    if idx < 0 || idx >= len(arr) {
        fmt.Println("Indeks tidak valid!")
        return arr
    }
    fmt.Printf("Menghapus elemen di indeks %d: %d\n", idx, arr[idx])
    return append(arr[:idx], arr[idx+1:]...)
}

func rataRata(arr []int) float64 {
    total := 0
    for _, val := range arr {
        total += val
    }
    return float64(total) / float64(len(arr))
}

func standarDeviasi(arr []int) float64 {
    rata := rataRata(arr)
    var total float64
    for _, val := range arr {
        total += math.Pow(float64(val)-rata, 2)
    }
    return math.Sqrt(total / float64(len(arr)))
}

func hitungFrekuensi(arr []int, target int) int {
    count := 0
    for _, val := range arr {
        if val == target {
            count++
        }
    }
    return count
}

func main() {

```

```

arr := []int{5, 10, 15, 20, 25, 30, 35, 40}

tampilkanArray(arr)

tampilkanIndeksGanjil(arr)

tampilkanIndeksGenap(arr)

var x int
fmt.Print("Masukkan kelipatan indeks yang diinginkan: ")
fmt.Scan(&x)
tampilkanKelipatanIndeks(arr, x)

var idx int
fmt.Print("Masukkan indeks yang ingin dihapus: ")
fmt.Scan(&idx)
arr = hapusElemen(arr, idx)
tampilkanArray(arr)

fmt.Printf("Rata-rata elemen array: %.2f\n", rataRata(arr))

fmt.Printf("Standar deviasi elemen array: %.2f\n", standarDeviasi(arr))

var target int
fmt.Print("Masukkan bilangan untuk menghitung frekuensinya: ")
fmt.Scan(&target)
fmt.Printf("Frekuensi bilangan %d: %d\n", target, hitungFrekuensi(arr,
target))
}

```

Sceenshoot Output

```

PS C:\Users\Syahrul\SEMESTER3\Modul7> go run "c:\Users\Syahrul\SEMESTER3\Modul7\Unguided2\main.go"
Isi array: [5 10 15 20 25 30 35 40]
Elemen dengan indeks ganjil: 10 20 30 40
Elemen dengan indeks genap: 5 15 25 35
Masukkan kelipatan indeks yang diinginkan: 2
Elemen dengan indeks kelipatan 2: 15 25 35
Masukkan indeks yang ingin dihapus: 3
Menghapus elemen di indeks 3: 20
Elemen dengan indeks kelipatan 2: 15 25 35
Masukkan indeks yang ingin dihapus: 3
Elemen dengan indeks kelipatan 2: 15 25 35
Elemen dengan indeks kelipatan 2: 15 25 35
Masukkan indeks yang ingin dihapus: 3
Menghapus elemen di indeks 3: 20
Elemen dengan indeks kelipatan 2: 15 25 35
Elemen dengan indeks kelipatan 2: 15 25 35
Masukkan indeks yang ingin dihapus: 3
Menghapus elemen di indeks 3: 20
Isi array: [5 10 15 25 30 35 40]
Rata-rata elemen array: 22.86
Standar deviasi elemen array: 12.21
Masukkan bilangan untuk menghitung frekuensinya: 15
Frekuensi bilangan 15: 1
PS C:\Users\Syahrul\SEMESTER3\Modul7>

```

Deskripsi Program

Program ini adalah aplikasi berbasis Golang yang menggunakan array untuk memproses sekumpulan bilangan bulat dan menyediakan berbagai operasi analisis. Program memungkinkan pengguna untuk menampilkan isi array, elemen-elemen pada indeks ganjil, genap, atau kelipatan tertentu, serta menghapus elemen pada indeks spesifik. Selain itu, program dapat menghitung rata-rata, standar deviasi, dan frekuensi kemunculan elemen tertentu dalam array. Program ini dirancang interaktif, di mana pengguna dapat memberikan input seperti indeks yang akan dihapus atau bilangan untuk dihitung frekuensinya, sehingga hasilnya dinamis dan sesuai dengan kebutuhan analisis data pengguna.

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 ber laga.

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.

Perhatikan sesi interaksi pada contoh berikut ini (teks bergaris bawah adalah input/read)

```
Klub A : MU
Klub B : Inter
Pertandingan 1 : 2    0           // MU = 2 sedangkan Inter = 0
Pertandingan 2 : 1    2
Pertandingan 3 : 2    2
Pertandingan 4 : 0    1
Pertandingan 5 : 3    2
Pertandingan 6 : 1    0
Pertandingan 7 : 5    2
Pertandingan 8 : 2    3
Pertandingan 9 : -1   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
```

Sourcecode

```
package main

import (
    "fmt"
)

type Klub struct {
    nama string
}

type Pertandingan struct {
    skorA int
    skorB int
}

func main() {
    var klubA, klubB Klub
```

```

var pertandingan []Pertandingan
var pemenang []string

fmt.Print("Masukkan nama Klub A: ")
fmt.Scanln(&klubA.nama)
fmt.Print("Masukkan nama Klub B: ")
fmt.Scanln(&klubB.nama)

fmt.Println("\nMasukkan skor pertandingan (format: skorA skorB)")

for {
    var skorA, skorB int
    fmt.Print("Skor: ")
    fmt.Scanf("%d %d", &skorA, &skorB)

    if skorA < 0 || skorB < 0 {
        break
    }

    pertandingan = append(pertandingan, Pertandingan{skorA, skorB})

    if skorA > skorB {
        pemenang = append(pemenang, klubA.nama)
    } else if skorA < skorB {
        pemenang = append(pemenang, klubB.nama)
    } else {
        pemenang = append(pemenang, "Draw")
    }
}

fmt.Println("\nHasil pertandingan:")
for i, p := range pertandingan {
    fmt.Printf("Pertandingan %d: %s %d - %d %s\n", i+1, klubA.nama,
p.skorA, p.skorB, klubB.nama)
}

fmt.Println("\nDaftar pemenang:")
for i, p := range pemenang {
    fmt.Printf("Hasil %d: %s\n", i+1, p)
}
}

```


Screenshoot

```
PS C:\Users\Syahru1\SEMESTER3\Modul7> go run "c:\Users\Syahru1\SEMESTER3\Modul7\Unguided3\main.go"
Masukkan nama Klub A: MU
Masukkan nama Klub B: Inter

Masukkan skor pertandingan (format: skorA skorB)
Skor: 2 0
Skor: Skor: 1 2
Skor: Skor: 2 2
Skor: Skor: 2 2
Skor: Skor: 0 1
Skor: Skor: 3 2
Skor: Skor: 1 0
Skor: Skor: 5 2
Skor: Skor: 2 3
Skor: Skor: -1 2

Hasil pertandingan:
Pertandingan 1: MU 2 - 0 Inter
Pertandingan 2: MU 0 - 0 Inter
Pertandingan 3: MU 1 - 2 Inter
Pertandingan 4: MU 0 - 0 Inter
Pertandingan 5: MU 2 - 2 Inter
Pertandingan 6: MU 0 - 0 Inter
Pertandingan 7: MU 2 - 2 Inter
Pertandingan 8: MU 0 - 0 Inter
Pertandingan 9: MU 0 - 1 Inter
Pertandingan 10: MU 0 - 0 Inter
Pertandingan 11: MU 3 - 2 Inter
Pertandingan 12: MU 0 - 0 Inter
Pertandingan 13: MU 1 - 0 Inter
Pertandingan 14: MU 0 - 0 Inter
Pertandingan 15: MU 5 - 2 Inter
Pertandingan 16: MU 0 - 0 Inter
Pertandingan 17: MU 2 - 3 Inter
Pertandingan 18: MU 0 - 0 Inter

Daftar pemenang:
Hasil 1: MU
Hasil 2: Draw
Hasil 3: Inter
Hasil 4: Draw
Hasil 5: Draw
Hasil 6: Draw
Hasil 7: Draw
Hasil 8: Draw
Hasil 9: Inter
Hasil 10: Draw
Hasil 11: MU
Hasil 12: Draw
Hasil 13: MU
Hasil 14: Draw
Hasil 15: MU
Hasil 16: Draw
Hasil 17: Inter
Hasil 18: Draw
```

Deskripsi Program

Program di atas adalah aplikasi berbasis Go untuk mencatat dan menampilkan hasil pertandingan antara dua klub sepak bola. Program dimulai dengan meminta pengguna memasukkan nama kedua klub (Klub A dan Klub B). Selanjutnya, program menerima input skor pertandingan secara berulang dalam format angka (skorA dan skorB). Proses input skor berlangsung hingga pengguna memasukkan skor negatif untuk salah satu atau kedua klub, yang menandakan akhir dari input. Berdasarkan skor yang diberikan, program menentukan pemenang untuk setiap pertandingan (Klub A, Klub B, atau "Draw" jika skor seri) dan menyimpannya dalam daftar pemenang. Di akhir eksekusi, program menampilkan daftar pertandingan lengkap dengan skor

masing-masing serta daftar pemenang dari setiap pertandingan. Program ini menggunakan **struct** untuk mendefinisikan data klub dan pertandingan serta **slice** untuk menyimpan data pertandingan dan hasil pemenang secara dinamis.

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.

Lengkapi potongan algoritma berikut ini!

```
package main
import "fmt"
const NMAX int = 127
type tabel [NMAX]rune
    tab : tabel
    m : integer

func isiArray(t *tabel, n *int)
/*I.S. Data tersedia dalam piranti masukan
  F.S. Array t berisi sejumlah n karakter yang dimasukkan user,
  Proses input selama karakter bukanlah TITIK dan n <= NMAX */
```

```

func cetakArray(t tabel, n int)
/*I.S. Terdefinisi array t yang berisi sejumlah n karakter
  F.S. n karakter dalam array muncul di layar */

func balikanArray(t *tabel, n int)
/*I.S. Terdefinisi array t yang berisi sejumlah n karakter
  F.S. Urutan isi array t terbalik */

func main(){
    var tab tabel
    var m int
    // si array tab dengan memanggil prosedur isiArray

    // Balikian isi array tab dengan memanggil balikanArray

    // Cetak isi array tab
}

```

Perhatikan sesi interaksi pada contoh berikut ini (teks bergaris bawah adalah input/read)

```

Teks      : S E N A N G ,
Reverse teks : G N A N E S

Teks      : K A T A K .
Reverse teks : K A T A K

```

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

***Palindrom adalah teks yang dibaca dari awal atau akhir adalah sama, contoh: KATAK, APA, KASUR_RUSAK.**

```

func palindrom(t tabel, n int) bool
/* Mengembalikan true apabila susunan karakter di dalam t membentuk palindrom,
  dan false apabila sebaliknya. Petunjuk: Manfaatkan prosedur balikanArray */

```

Perhatikan sesi interaksi pada contoh berikut ini (teks bergaris bawah adalah input/read)

```

Teks      : K A T A K
Palindrom : ? true

Teks      : S E N A N G
Palindrom : ? false

```

Sourcecode

```

package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

```

```

const NMAX int = 127

type Tabel struct {
    tab [NMAX]rune
    m   int
}

func isiArray(t *Tabel) {
    reader := bufio.NewReader(os.Stdin)
    fmt.Println("Masukkan teks (akhiri dengan '.'): ")
    input, _ := reader.ReadString('.')
    input = strings.TrimSuffix(input, ".")

    for i, r := range input {
        if i >= NMAX {
            break
        }
        t.tab[i] = r
        t.m++
    }
}

func cetakArray(t Tabel) {
    fmt.Print("Array: ")
    for i := 0; i < t.m; i++ {
        fmt.Print(string(t.tab[i]))
    }
    fmt.Println()
}

func balikkanArray(t *Tabel) {
    for i, j := 0, t.m-1; i < j; i, j = i+1, j-1 {
        t.tab[i], t.tab[j] = t.tab[j], t.tab[i]
    }
}

func palindrome(t Tabel) bool {
    for i := 0; i < t.m/2; i++ {
        if t.tab[i] != t.tab[t.m-1-i] {
            return false
        }
    }
    return true
}

func main() {
    var tab Tabel

    isiArray(&tab)

```

```

    fmt.Println("\nArray asli:")
    cetakArray(tab)

    isPalindrome := palindrome(tab)
    if isPalindrome {
        fmt.Println("Palindrome: true")
    } else {
        fmt.Println("Palindrome: false")
    }

    balikkanArray(&tab)
    fmt.Println("\nArray setelah dibalik:")
    cetakArray(tab)
}

```

Screenshoot

```

PS C:\Users\Syahrul\SEMESTER3\Modul17> go run "c:\Users\Syahrul\SEMESTER3\Modul17\Unguided4\main.go"
Masukkan teks (akhiri dengan '.'):
KATAK'.'
Array asli:
Array asli:
Array: KATAK'
Palindrome: false
Array setelah dibalik:
Array: 'KATAK'
PS C:\Users\Syahrul\SEMESTER3\Modul17>

```

Deskripsi Program

Program di atas adalah aplikasi Go yang memanfaatkan **struct** dan **array** untuk mengolah teks dan memeriksa apakah teks tersebut merupakan palindrome. Program dimulai dengan meminta pengguna memasukkan teks, yang akan disimpan dalam sebuah array hingga menemukan karakter titik (.). Teks yang dimasukkan kemudian dicetak, dan program memeriksa apakah teks tersebut palindrome dengan membandingkan elemen dari awal hingga tengah array secara berpasangan dengan elemen dari akhir. Selanjutnya, program membalikkan urutan teks dalam array menggunakan teknik *two-pointer* dan mencetak hasilnya. Program ini dirancang untuk menangani teks dengan panjang maksimum yang ditentukan, memastikan efisiensi dan validitas proses pengolahan teks.

KESIMPULAN

Golang menyediakan dukungan yang kuat untuk struct dan array sebagai cara untuk mengelola data secara terstruktur dan efisien. Struct digunakan untuk mendefinisikan tipe data kustom yang dapat berisi berbagai atribut, memungkinkan pengelolaan data yang kompleks dalam bentuk yang lebih terorganisir. Array, di sisi lain, digunakan untuk menyimpan elemen dengan tipe data yang sama dalam jumlah tetap, menjadikannya ideal untuk pengolahan data yang membutuhkan akses cepat dan langsung ke elemen berdasarkan indeks.

Dalam implementasi program, kombinasi struct dan array mempermudah pengelolaan data berukuran besar atau terorganisir, seperti tabel, list, atau struktur data lain. Dengan array, pengolahan data seperti pencarian, pembalikan, dan pemeriksaan pola (seperti palindrome) dapat dilakukan secara efisien. Penggunaan pointer dalam fungsi memungkinkan manipulasi data secara langsung tanpa menduplikasi memori, meningkatkan efisiensi.

Secara keseluruhan, pemahaman dan penggunaan struct dan array di Go memungkinkan pengembangan aplikasi yang lebih modular, terstruktur, dan efisien, terutama untuk kasus yang membutuhkan pengolahan data secara sistematis.

DAFTAR PUSTAKA

<https://jocodev.id/struktur-data-di-go-pemahaman-dan-implementasi/>