

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL VII  
STRUCT & ARRAY**



**Disusun Oleh :**

**Wafiq Nur Azizah / 2311102270**

**S1IF-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. Tipe Bentuk (Type)

Dalam bahasa go, pengguna dapat mendefinisikan tipe data baru menggunakan keyword 'type'. Tipe ini bisa berupa alias dari tipe yang sudah ada atau bisa juga tipe data yang lebih kompleks seperti struct, interface, dan lain sebagainya. Tipe bentukan ini sangat berguna ketika ingin memberi nama yang lebih jelas pada tipe tertentu dalam program sehingga kode menjadi lebih mudah dibaca dan dipahami.

```
package main

import "fmt"

// Alias tipe data untuk umur
type Age int

func main() {
    var myAge Age = 30 // Menggunakan tipe Age sebagai
    alias int
    fmt.Println(myAge)
}
```

### B. Struct

Struct adalah tipe data yang digunakan untuk mengelompokkan beberapa nilai dengan tipe yang berbeda. Setiap elemen dalam struct disebut sebagai field. Struct sangat berguna untuk mendefinisikan objek yang memiliki beberapa atribut terkait. Contoh penerapan yang umum adalah pada pembuatan aplikasi berbasis objek, seperti yang dapat ditemui di bahasa pemrograman berorientasi objek lainnya.

```
package main

import "fmt"

// Mendefinisikan struct
type Person struct {
    Name    string
    Age     int
    Address string
}

func main() {
    // Membuat instance struct Person
    p := Person{Name: "Alice", Age: 30, Address: "123
Main St"}
```

```

// Mengakses field dari struct
fmt.Println("Name:", p.Name)
fmt.Println("Age:", p.Age)
fmt.Println("Address:", p.Address)

// Mengubah field struct
p.Age = 31
fmt.Println("Updated Age:", p.Age)
}

```

### C. Array dan Slice (Array Dinamis)

Array di Go adalah koleksi elemen dengan tipe data yang sama dan ukuran yang tetap. Array sangat berguna ketika tahu jumlah elemen yang tetap di awal program. Namun, jika jumlah elemen berubah-ubah, maka array bukan pilihan terbaik dan lebih baik menggunakan slice. Slice adalah tipe data yang lebih fleksibel dibandingkan array karena slice memiliki panjang yang dapat berubah-ubah. Slice adalah representasi dari bagian dari array sehingga pengguna dapat bagian dari array yang lebih besar tanpa menyalinnya.

```

package main

import "fmt"

func main() {
    // Membuat slice dengan elemen awal
    slice := []int{1, 2, 3}

    // Menambah elemen ke slice
    slice = append(slice, 4, 5)

    // Mencetak slice
    fmt.Println(slice) // Output: [1 2 3 4 5]

    // Mengakses elemen slice
    fmt.Println("First element:", slice[0]) // Output:
    First element: 1
}

```

#### D. Map

Map adalah koleksi pasangan key-value yang memungkinkan pengguna untuk menyimpan data dalam bentuk asosiasi antara kunci (key) dan nilai (value). Map sangat berguna ketika perlu mengakses data dengan cara yang cepat berdasarkan kunci unik, seperti dalam database atau pencarian data yang efisien.

```
package main

import "fmt"

func main() {
    // Mendeklarasikan map dengan key bertipe string dan
    // value bertipe int
    m := make(map[string]int)

    // Menambahkan pasangan key-value ke map
    m["age"] = 25
    m["score"] = 100

    // Mengakses nilai berdasarkan key
    fmt.Println("Age:", m["age"]) // Output: Age: 25
    fmt.Println("Score:", m["score"]) // Output: Score:
100

    // Menghapus pasangan key-value dari map
    delete(m, "score")

    // Mencetak map setelah penghapusan
    fmt.Println("Map after deletion:", m) // Output: Map
after deletion: map[age:25]
}
```

## II. GUIDED

### 1. Soal Studi Case

#### Sourcecode

```
package main

import "fmt"

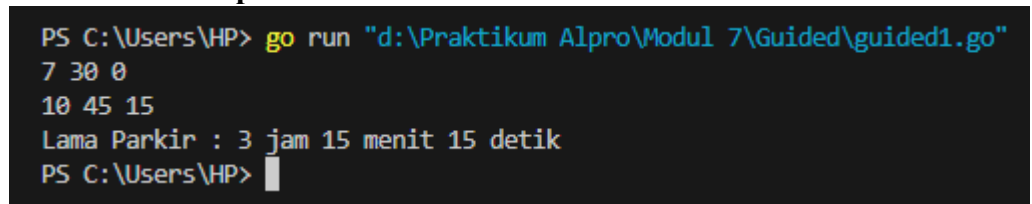
type waktu struct {
    jam, menit, detik int
}

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
    dPulang = wPulang.detik + wPulang.menit*60 +
wPulang.jam*3600
    lParkir = dPulang - dParkir

    durasi.jam = lParkir / 3600
    durasi.menit = lParkir % 3600 / 60
    durasi.detik = lParkir % 3600 % 60
    fmt.Printf("Lama Parkir : %d jam %d menit %d detik",
durasi.jam, durasi.menit, durasi.detik)
}
```

#### Screenshoot Output



```
PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 7\Guided\guided1.go"
7 30 0
10 45 15
Lama Parkir : 3 jam 15 menit 15 detik
PS C:\Users\HP>
```

#### Deskripsi Program

Program ini menghitung durasi parkir kendaraan berdasarkan waktu masuk dan keluar yang diinput pengguna dengan memanfaatkan tipe data struct bernama `waktu` untuk merepresentasikan jam, menit, dan detik. Input waktu parkir (`wParkir`) dan waktu pulang (`wPulang`) dikonversi menjadi total detik menggunakan formula `total detik = jam\*3600 + menit\*60 + detik` untuk mempermudah perhitungan. Selisih total detik

kemudian diubah kembali ke format jam, menit, dan detik melalui operasi pembagian dan sisa pembagian. Hasil akhirnya berupa durasi parkir dalam format "X jam Y menit Z detik". Cara ini mengoptimalkan perhitungan selisih waktu dengan memanfaatkan konversi ke satuan detik serta penggunaan tipe data terstruktur untuk pengelolaan data yang lebih sistematis.

## 2. Soal Studi Case

### 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\HP> go run "d:\Praktikum Alpro\Modul 7\Guided\guided2.go"
Nama Budi sudah ada dalam daftar.
Daftar Teman: [Andi Budi Cici Dewi Eka]
PS C:\Users\HP>
```

### Deskripsi Program

Program sederhana dalam bahasa Go ini memperbarui daftar teman dengan nama-nama baru sekaligus memeriksa keberadaan nama tersebut dalam daftar menggunakan fungsi `sudahAda`. Program dimulai dengan mendefinisikan `daftarTeman` sebagai slice berisi nama-nama awal, lalu memproses `namaBaru`. Slice yang berisi nama-nama teman baru melalui loop. Dalam setiap iterasi, fungsi `sudahAda` memeriksa apakah nama tersebut sudah ada di `daftarTeman`. Jika belum, nama ditambahkan dan jika sudah, maka program akan mencetak pesan bahwa nama tersebut sudah ada. Setelah semua nama diproses, program mencetak daftar teman yang telah diperbarui, menggambarkan penggunaan slice, fungsi, dan pengendalian alur dengan `if-else`.

### 3. Soal Studi Case

#### Sourcecode

```
package main

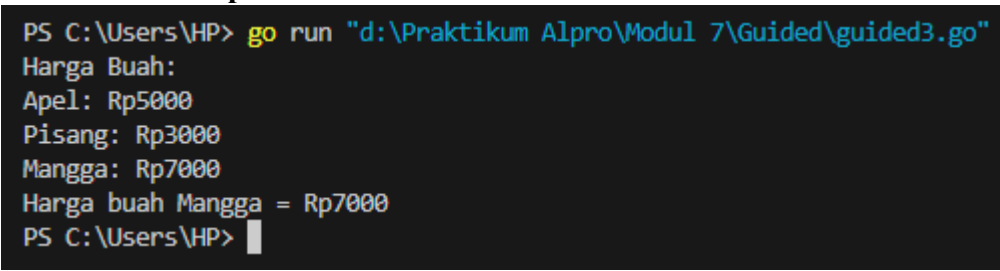
import "fmt"

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.Printf("Harga buah Mangga = Rp%d\n",
        hargaBuah["Mangga"])
}
```

#### Screenshot Output



```
PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 7\Guided\guided3.go"
Harga Buah:
Apel: Rp5000
Pisang: Rp3000
Mangga: Rp7000
Harga buah Mangga = Rp7000
PS C:\Users\HP>
```

### **Deskripsi Program**

Program go di atas adalah implementasi sederhana menggunakan struktur data map untuk menyimpan pasangan key-value, dimana nama buah sebagai key dan harga buah sebagai value. Dalam map `hargaBuah`, terdapat tiga jenis buah, yaitu apel, pisang, dan mangga beserta harga masing-masing. Program pertama-tama mencetak semua nama buah dan harga melalui perulangan `for range`, lalu mengakses harga mangga secara langsung dengan kunci `hargaBuah["Mangga"]`. Program ini menunjukkan cara mengakses data secara spesifik dan efisien menggunakan map.



### III. UNGUIDED

#### 1. Soal Studi Case

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:

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

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(t1, t2 titik) float64 {
    return math.Sqrt(math.Pow(float64(t1.x-t2.x), 2) +
        math.Pow(float64(t1.y-t2.y), 2))
}

func lokasititik(t titik, l lingkaran) bool {
    return jarak(titik{t.x, t.y}, titik{l.cx, l.cy}) <=
        float64(l.r)
}

func main() {
    var l1, l2 lingkaran
    var t titik

    fmt.Print("Masukkan Titik Pusat Lingkaran 1 (cx cy): ")
    fmt.Scan(&l1.cx, &l1.cy)
    fmt.Print("Masukkan Radius Lingkaran 1: ")
    fmt.Scan(&l1.r)
    fmt.Print("Masukkan Titik Pusat Lingkaran 2 (cx cy): ")
    fmt.Scan(&l2.cx, &l2.cy)
    fmt.Print("Masukkan Radius Lingkaran 2: ")
    fmt.Scan(&l2.r)
```

```

        fmt.Print("Masukkan Koordinat Titik (x y): ")
        fmt.Scan(&t.x, &t.y)

        if lokasititik(t, l1) && lokasititik(t, l2) {
            fmt.Println("Titik di dalam lingkaran 1 dan
2")
        } else if lokasititik(t, l1) {
            fmt.Println("Titik di dalam lingkaran 1")
        } else if lokasititik(t, l2) {
            fmt.Println("Titik di dalam lingkaran 2")
        } else {
            fmt.Println("Titik di luar lingkaran 1 dan 2")
        }
    }
}

```

### Screenshoot Output

```

PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 7\Unguided\unguided1.go"
Masukkan Titik Pusat Lingkaran 1 (cx cy): 0 0
Masukkan Radius Lingkaran 1: 5
Masukkan Titik Pusat Lingkaran 2 (cx cy): 4 4
Masukkan Radius Lingkaran 2: 3
Masukkan Koordinat Titik (x y): 2 2
Titik di dalam lingkaran 1 dan 2
PS C:\Users\HP> 

```

### Deskripsi Program

Program ini bertujuan untuk menentukan apakah sebuah titik berada di dalam satu atau kedua lingkaran dengan menggunakan dua struktur data, yaitu `titik` untuk koordinat titik (x, y) dan `lingkaran` untuk pusat (cx, cy) serta radius (r) lingkaran. Fungsi `jarak` menghitung jarak antara dua titik menggunakan rumus Pythagoras, sementara fungsi `lokasititik` memeriksa apakah titik berada dalam lingkaran dengan membandingkan jarak titik ke pusat lingkaran dengan radiusnya. Dalam bagian utama program, input dari pengguna untuk dua lingkaran dan satu titik akan diperiksa, kemudian program menentukan apakah titik tersebut berada dalam lingkaran pertama, kedua, atau keduanya, dan menampilkan hasilnya. Program ini menggambarkan konsep dasar geometri dalam pemrograman, seperti penghitungan jarak dan pengujian kondisi dalam lingkaran.

## 2. Soal Studi Case

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 dihapus 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 main() {
    var jumlahElemen int
    fmt.Print("Masukkan jumlah elemen array: ")
    fmt.Scan(&jumlahElemen)
    dataArray := make([]int, jumlahElemen)

    for i := 0; i < jumlahElemen; i++ {
        fmt.Print("Masukkan nilai untuk elemen array ke-", i, ": ")
        fmt.Scan(&dataArray[i])
    }

    menuPilihan(dataArray)
}

func menuPilihan(array []int) {
    var pilihanMenu int
    n := len(array)
```

```

        fmt.Println("-----MENU PENGOLAHAN DATA ARRAY-----")
        fmt.Println("1. Tampilkan seluruh isi array")
        fmt.Println("2. Tampilkan elemen array dengan indeks ganjil")
        fmt.Println("3. Tampilkan elemen array dengan indeks genap")
        fmt.Println("4. Tampilkan elemen array yang indeksinya kelipatan x")
        fmt.Println("5. Hapus elemen array pada indeks tertentu")
        fmt.Println("6. Tampilkan rata-rata elemen array")
        fmt.Println("7. Tampilkan simpangan baku array")
        fmt.Println("8. Tampilkan frekuensi suatu bilangan dalam array")
        fmt.Print("Masukkan pilihan Anda: ")
        fmt.Scan(&pilihanMenu)

        switch pilihanMenu {
        case 1:
            for i := 0; i < n; i++ {
                fmt.Print(array[i], " ")
            }
            fmt.Println()
        case 2:
            for i := 1; i < n; i += 2 {
                fmt.Print(array[i], " ")
            }
            fmt.Println()
        case 3:
            for i := 0; i < n; i += 2 {
                fmt.Print(array[i], " ")
            }
            fmt.Println()
        case 4:
            var kelipatanX int
            fmt.Print("Masukkan nilai kelipatan (x): ")
            fmt.Scan(&kelipatanX)
            for i := 0; i < n; i++ {
                if i%kelipatanX == 0 {
                    fmt.Print(array[i], " ")
                }
            }
            fmt.Println()
        case 5:
            var indeksHapus int
            fmt.Print("Masukkan indeks yang ingin dihapus: ")

            fmt.Scan(&indeksHapus)
            if indeksHapus < 0 || indeksHapus >= n {
                fmt.Println("Indeks tidak valid!")
                return
            }

```

```

    }
    arrayBaru := make([]int, 0, n-1)
    for i := 0; i < n; i++ {
        if i != indeksHapus {
            arrayBaru = append(arrayBaru, array[i])
        }
    }
    fmt.Println("Array setelah penghapusan:",
arrayBaru)
    case 6:
        var jumlah int
        for i := 0; i < n; i++ {
            jumlah += array[i]
        }
        rataRata := float64(jumlah) / float64(n)
        fmt.Println("Rata-rata elemen array adalah:",
rataRata)
    case 7:
        var selisih, totalSelisih float64
        var jumlah int
        for i := 0; i < n; i++ {
            jumlah += array[i]
        }
        rataRata := float64(jumlah) / float64(n)
        for i := 0; i < n; i++ {
            selisih = float64(array[i]) - rataRata
            totalSelisih += selisih * selisih
        }
        simpanganBaku := math.Sqrt(totalSelisih /
float64(n))
        fmt.Println("Simpangan baku dari array adalah:",
simpanganBaku)
    case 8:
        var frekuensi, angka int
        fmt.Print("Masukkan angka yang frekuensinya
ingin dihitung: ")
        fmt.Scan(&angka)
        for i := 0; i < n; i++ {
            if array[i] == angka {
                frekuensi++
            }
        }
        fmt.Printf("Frekuensi bilangan %d dalam array
adalah %d\n", angka, frekuensi)
    default:
        fmt.Println("Pilihan tidak valid!")
    }
}

```

## Screenshoot Output

```
Masukkan nilai untuk elemen array ke-1: 6
Masukkan nilai untuk elemen array ke-2: 9
Masukkan nilai untuk elemen array ke-3: 12
-----MENU PENGOLAHAN DATA ARRAY-----
1. Tampilkan seluruh isi array
2. Tampilkan elemen array dengan indeks ganjil
3. Tampilkan elemen array dengan indeks genap
4. Tampilkan elemen array yang indeksnya kelipatan x
5. Hapus elemen array pada indeks tertentu
6. Tampilkan rata-rata elemen array
7. Tampilkan simpangan baku array
8. Tampilkan frekuensi suatu bilangan dalam array
Masukkan pilihan Anda: 6
Rata-rata elemen array adalah: 7.5
PS C:\Users\HP> █
```

## Deskripsi Program

Program go di atas adalah untuk mengolah dan menganalisis data dalam bentuk array. Pengguna diminta memasukkan jumlah elemen dan nilai-nilai array, kemudian memilih pilihan operasi yang diinginkan seperti menampilkan array, menampilkan elemen dengan indeks ganjil/genap, menghapus elemen, menghitung rata-rata, simpangan baku, dan frekuensi kemunculan angka. Fungsi utama, `menuPilihan` akan menjalankan operasi sesuai pilihan pengguna. Variabel penting seperti `dataArray` menyimpan elemen array, `jumlahElemen` menyimpan jumlah elemen, dan `pilihanMenu` mengarahkan program ke operasi yang dipilih. Hasil dari setiap operasi kemudian akan ditampilkan di layar.

### 3. Soal Studi Case

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 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 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"

func main() {
    var klubA, klubB string
    var skorA, skorB int
    var hasil []string

    fmt.Print("Klub A: ")
    fmt.Scan(&klubA)
    fmt.Print("Klub B: ")
    fmt.Scan(&klubB)

    i := 1
    for {
        fmt.Printf("Pertandingan %d: ", i)
        fmt.Scan(&skorA, &skorB)

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

        if skorA > skorB {
            hasil = append(hasil, klubA)
        } else if skorA < skorB {
            hasil = append(hasil, klubB)
        } else {
            hasil = append(hasil, "Seri")
        }
        i++
    }

    fmt.Println("Daftar klub yang memenangkan pertandingan:")
    for j := 0; j < len(hasil); j++ {
        fmt.Printf("Pertandingan %d: %s\n", j+1, hasil[j])
    }

    fmt.Println("Pertandingan selesai")
}
```

## Screenshoot Output

```
PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 7\Unguided\unguided3.go"
Klub A: MU
Klub B: Inter
Pertandingan 1: 2 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
Daftar klub yang memenangkan pertandingan:
Pertandingan 1: MU
Pertandingan 2: Inter
Pertandingan 3: Seri
Pertandingan 4: Inter
Pertandingan 5: MU
Pertandingan 6: MU
Pertandingan 7: MU
Pertandingan 8: Inter
Pertandingan selesai
PS C:\Users\HP> █
```

## Deskripsi Program

Program go di atas adalah untuk mencatat hasil pertandingan antara dua klub sepak bola yaitu, Klub A dan Klub berdasarkan skor yang diinput pengguna. Pengguna diminta memasukkan nama kedua klub dan skor masing-masing tim secara berulang menggunakan perulangan `for`. Hasil pertandingan dicatat dalam slice `hasil` yang menentukan pemenangnya (Klub A, Klub B, atau "Seri"). Perulangan berhenti ketika skor negatif dimasukkan untuk salah satu tim, dan program kemudian menampilkan daftar pemenang dari setiap pertandingan.

Bagian utama program meliputi variabel untuk menyimpan nama klub (`klubA`, `klubB`) dan skor tim (`skorA`, `skorB`), serta slice `hasil` untuk mencatat hasil pertandingan. Fungsi seperti `fmt.Scan` digunakan untuk mengambil input, `append` untuk menambahkan data ke slice, dan `fmt.Printf` atau `fmt.Println` untuk menampilkan hasil. Seluruh logika program dijalankan dalam fungsi `main` sebagai titik awal eksekusi program.

#### 4. Study Case

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 is 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 I A K
Palindrom  ? true

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

### Sourcecode

```
package main

import (
    "fmt"
    "unicode"
)

const MAX_SIZE int = 127

type charArray [MAX_SIZE]rune

func fillArray(arr *charArray, size *int) {
    fmt.Println("Masukkan karakter (ketik '.' untuk berhenti):")
    for *size < MAX_SIZE {
        var input rune
        fmt.Scanf("%c", &input)
        if unicode.IsSpace(input) {
            continue
        }
        if input == '.' {
            break
        }
        arr[*size] = unicode.ToLower(input)
        (*size)++
    }
}

func printArray(arr charArray, size int) {
    for i := 0; i < size; i++ {
        fmt.Printf("%c", arr[i])
    }
    fmt.Println()
}

func reverseArray(original charArray, reversed
*charArray, size int) {
    for i := 0; i < size; i++ {
        reversed[i] = original[size-1-i]
```

```

    }
}

func isPalindrome(original charArray, reversed
charArray, size int) bool {
    for i := 0; i < size; i++ {
        if original[i] != reversed[i] {
            return false
        }
    }
    return true
}

func main() {
    var inputArray charArray
    var reversedArray charArray
    var arraySize int = 0

    fillArray(&inputArray, &arraySize)
    fmt.Print("Isi array: ")
    printArray(inputArray, arraySize)

    reverseArray(inputArray, &reversedArray, arraySize)
    fmt.Print("Array setelah dibalik: ")
    printArray(reversedArray, arraySize)

    if isPalindrome(inputArray, reversedArray,
arraySize) {
        fmt.Println("Array adalah palindrom")
    } else {
        fmt.Println("Array bukan palindrom")
    }
}

```

## Screenshoot Output

```

Masukkan karakter (ketik '.' untuk berhenti):
katak.
Isi array: katak
Array setelah dibalik: katak
Array adalah palindrom
PS C:\Users\HP> go run "d:\Praktikum Alpro\Modul 7\Unguided\unguided4.go"
Masukkan karakter (ketik '.' untuk berhenti):
senang.
Isi array: senang
Array setelah dibalik: gnanes
Array bukan palindrom
PS C:\Users\HP> 

```

### **Deskripsi Program**

Program go di atas digunakan untuk memeriksa apakah sebuah string merupakan palindrom, yaitu kata yang sama jika dibaca dari depan maupun belakang, seperti "katak". Program menerima input karakter dari pengguna hingga batas tertentu atau hingga pengguna mengetikkan karakter `.` untuk berhenti. Input yang diberikan akan diubah menjadi huruf kecil dan mengabaikan spasi untuk memudahkan pemeriksaan. Terdapat beberapa fungsi utama dalam program ini: `fillArray` untuk mengisi array dengan input yang sudah diproses, `reverseArray` untuk membalik urutan elemen array, `isPalindrome` untuk membandingkan array asli dengan yang dibalik, dan `printArray` untuk menampilkan hasilnya. Program ini menggunakan array `inputArray` untuk menyimpan input, `reversedArray` untuk hasil pembalikan, dan `arraySize` untuk melacak jumlah elemen yang valid, dengan batas maksimal input 127 karakter.

#### IV. DAFTAR PUSTAKA

- Agisti, T. (2024, November 17). Pengenalan operasi map di Golang. Medium. Diakses pada 17 November 2024, dari <https://medium.com/@tiar.agisti/pengenalan-operasi-map-di-golang-83ff236d7cad>
- Golang ID. (2024, November 17). Pemahaman tentang Slices di Go. Golang ID. Diakses pada 17 November 2024, dari <https://golang-id.org/blog/slices/>
- Novalagung. (n.d.). A Tipe Data dalam Pemrograman Go. Dasar Pemrograman Golang. Diakses pada 17 November 2024, dari <https://dasarpemrogramangolang.novalagung.com/A-tipe-data.html>
- Udacity. (2023, Juni 12). Go Golang Structs: Introduction, Syntax, and Examples. Udacity. Diakses pada 17 November 2024, dari <https://www.udacity.com/blog/2023/06/go-golang-structs.html>