

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 7
STRUCK & ARRAY**



Oleh:

NAMA: Lutfi Shidqi Mardian

NIM: 103112400077

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

I. DASAR TEORI

Tipe Bentuk (Struct dan Alias)

Dalam pemrograman Go, tipe bentuk digunakan untuk membuat tipe data baru yang lebih kompleks dari tipe data dasar. Tipe bentuk dibedakan menjadi dua jenis utama:

- **Alias (Type)**

Alias adalah pemberian nama baru untuk tipe data yang sudah ada, sehingga lebih mudah dibaca atau digunakan. Dalam Go, alias didefinisikan menggunakan kata kunci `type`.

Contoh:

```
type bilangan int  
  
type pecahan float64
```

Dengan cara ini, tipe `int` dan `float64` bisa diakses menggunakan nama baru yang lebih spesifik.

- **Struct (Structure/Record)**

Struct adalah kumpulan beberapa variabel yang memiliki hubungan, digabung menjadi satu kesatuan. Setiap elemen di dalam struct disebut field.

Contoh deklarasi struct di Go:

```
type waktu struct {  
    jam, menit, detik int  
}
```

Struct berguna untuk mengelompokkan data yang berkaitan, seperti data waktu, koordinat, atau informasi lainnya.

- **Array**

Array adalah kumpulan elemen dengan tipe data yang sama dan jumlah elemen yang tetap (statis) selama program berjalan. Di Go, deklarasi array menentukan jumlah elemen secara eksplisit.

Contoh deklarasi array:

```
var arr [5]int  
var buf = [5]byte{7, 3, 5, 2, 11}
```

Beberapa hal penting tentang array:

1. Indeks array di Go dimulai dari 0.
2. Ukuran array bisa diperiksa menggunakan fungsi `len(array)`.
3. Elemen dapat diakses dan dimodifikasi menggunakan indeks, misalnya `arr[0] = 10`.

- **Slice**

Slice adalah tipe data di Go yang mirip array tetapi memiliki ukuran yang bisa berubah selama eksekusi program. Slice lebih fleksibel dibandingkan array biasa. Slice bisa dibuat dari array, slice lain, atau menggunakan fungsi `make`.

Contoh deklarasi slice:

```
var sl = []int{1, 2, 3, 4}

var sl2 = make([]int, 10, 20)
```

Beberapa fungsi yang umum digunakan pada slice:

1. `len(slice)`: Mengembalikan jumlah elemen dalam slice.
2. `cap(slice)`: Mengembalikan kapasitas maksimum slice.
3. `append(slice, elemen)`: Menambahkan elemen baru ke slice

mfjmfj

- **Map**

Map adalah tipe data asosiatif di Go yang menyimpan pasangan key-value. Tidak seperti array, indeks pada map bisa berupa tipe data apapun (string, integer, float, dll).

Contoh deklarasi map:

```
var dct map[string]int

dct = map[string]int{"john": 25, "anne": 30}
```

II. GUIDED

1.

```
package main

import (
    "fmt"
    "time"
)

// Struct untuk barang dalam struk belanja
type Item struct {
    Name      string
    Price     float64
    Quantity  int
}

// Struct untuk struk belanja
type Receipt struct {
    StoreInfo string
    Date       time.Time
    Items      []Item
    TotalAmount float64
}

// Method untuk menghitung total harga semua item
func (r *Receipt) CalculateTotal() {
    var total float64
    for _, item := range r.Items {
        total += item.Price * float64(item.Quantity)
    }
}
```

```

    r.TotalAmount = total
}

// Method untuk mencetak struk belanja
func (r Receipt) PrintReceipt() {
    fmt.Println("=====
")
    fmt.Println(r.StoreInfo)
    fmt.Println("Tanggal:", r.Date.Format("02-01-2006
15:04"))
    fmt.Println("=====
")
    fmt.Printf("%-15s %-10s %-8s %-10s\n", "Item", "Harga",
    "Jumlah", "Total")
    fmt.Println("-----
")
    for _, item := range r.Items {
        itemTotal := item.Price * float64(item.Quantity)
        fmt.Printf("%-15s Rp%-9.2f %-8d Rp%-9.2f\n",
        item.Name, item.Price, item.Quantity, itemTotal)
    }
    fmt.Println("=====
")
    fmt.Printf("%-35s Rp%-9.2f\n", "Total Belanja:",
    r.TotalAmount)
    fmt.Println("=====
")
    fmt.Println("Terima kasih telah berbelanja!")
}

```

```
func main() {  
    receipt := Receipt{  
        StoreInfo: "Toko Sembako Makmur\nJl. Raya No. 123,  
Jakarta",  
        Date:      time.Now(),  
        Items: []Item{  
            {Name: "Beras", Price: 12000, Quantity: 5},  
            {Name: "Gula", Price: 15000, Quantity: 2},  
            {Name: "Minyak", Price: 20000, Quantity: 1},  
            {Name: "Telur", Price: 2000, Quantity: 10},  
        },  
    }  
    receipt.CalculateTotal()  
    receipt.PrintReceipt()  
}
```

Output Screenshot:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO02LUTFI> go run
=====
Toko Sembako Makmur
Jl. Raya No. 123, Jakarta
Tanggal: 27-04-2025 17:12
=====
Item          Harga      Jumlah  Total
-----
Beras         Rp12000.00    5      Rp60000.00
Gula          Rp15000.00    2      Rp30000.00
Minyak        Rp20000.00    1      Rp20000.00
Telur         Rp2000.00    10     Rp20000.00
=====
Total Belanja:                      Rp130000.00
=====
Terima kasih telah berbelanja!
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO02LUTFI> 
```

Penjelasan:

Program ini dirancang untuk mensimulasikan pembuatan struk belanja menggunakan bahasa pemrograman Go. Di dalamnya, terdapat dua struct utama, yaitu Item yang berisi nama, harga, dan jumlah setiap barang, serta Receipt yang menyimpan informasi toko, tanggal transaksi, daftar barang yang dibeli, dan total belanja keseluruhan. Dengan menggunakan method CalculateTotal, program menghitung total harga dari semua barang yang dibeli berdasarkan harga satuan dan jumlahnya. Setelah itu, struk belanja dicetak dengan rapi melalui method PrintReceipt, yang menampilkan rincian harga setiap barang, jumlah yang dibeli, serta total keseluruhan dengan format yang mudah dibaca. Program ini bertujuan untuk memberikan gambaran tentang bagaimana sistem struk belanja sederhana dapat diimplementasikan dalam bahasa Go.

2.

```
package main

import (
    "fmt"
)

func main() {
    // Deklarasi dan inisialisasi array nilai mahasiswa
    nilaiMahasiswa := [5]int{85, 90, 78, 88, 95}

    fmt.Println("Data Nilai Mahasiswa:")
    fmt.Println("=====")

    // Menampilkan nilai per mahasiswa
    for i, nilai := range nilaiMahasiswa {
        fmt.Printf("Mahasiswa %d: %d\n", i+1, nilai)
    }

    // Menghitung rata-rata nilai
    var total int

    for _, nilai := range nilaiMahasiswa {
        total += nilai
    }

    rataRata := float64(total) /
float64(len(nilaiMahasiswa))

    fmt.Println("=====")
    fmt.Printf("Rata-rata nilai: %.2f\n", rataRata)
```



```

// Mencari nilai tertinggi dan terendah
tertinggi := nilaiMahasiswa[0]
terendah := nilaiMahasiswa[0]
for _, nilai := range nilaiMahasiswa {
    if nilai > tertinggi {
        tertinggi = nilai
    }
    if nilai < terendah {
        terendah = nilai
    }
}

fmt.Printf("Nilai tertinggi: %d\n", tertinggi)
fmt.Printf("Nilai terendah: %d\n", terendah)

// Contoh array 2 dimensi
fmt.Println("\nContoh Array 2 Dimensi:")
fmt.Println("=====")

// Nilai ujian mahasiswa dalam 2 mata kuliah
(Matematika, Bahasa)

nilaiUjian := [3][2]int{
    {80, 85},
    {90, 75},
    {70, 95},
}

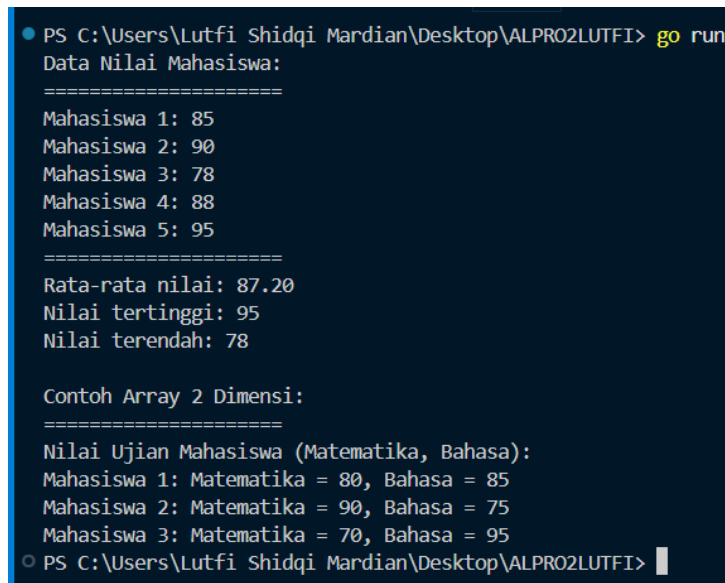
// Menampilkan nilai ujian per mahasiswa
fmt.Println("Nilai Ujian Mahasiswa (Matematika,
Bahasa):")

for i, nilai := range nilaiUjian {

```

```
        fmt.Printf("Mahasiswa %d: Matematika = %d, Bahasa = %d\n", i+1, nilai[0], nilai[1])
    }
}
```

Output Screenshot:



```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
Data Nilai Mahasiswa:
=====
Mahasiswa 1: 85
Mahasiswa 2: 90
Mahasiswa 3: 78
Mahasiswa 4: 88
Mahasiswa 5: 95
=====
Rata-rata nilai: 87.20
Nilai tertinggi: 95
Nilai terendah: 78

Contoh Array 2 Dimensi:
=====
Nilai Ujian Mahasiswa (Matematika, Bahasa):
Mahasiswa 1: Matematika = 80, Bahasa = 85
Mahasiswa 2: Matematika = 90, Bahasa = 75
Mahasiswa 3: Matematika = 70, Bahasa = 95
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

Penjelasan:

Program ini mendemonstrasikan penggunaan array satu dimensi dan dua dimensi dalam bahasa Go untuk menyimpan dan memproses data nilai mahasiswa. Array pertama (nilaiMahasiswa) berisi nilai lima mahasiswa, yang kemudian ditampilkan bersama dengan perhitungan rata-rata nilai, serta pencarian nilai tertinggi dan terendah. Program ini juga menunjukkan contoh penggunaan array dua dimensi untuk menyimpan nilai ujian mahasiswa dalam dua mata kuliah, yaitu Matematika dan Bahasa. Nilai-nilai tersebut ditampilkan dengan format yang rapi, serta menampilkan informasi per mahasiswa. Program ini memberikan gambaran yang jelas tentang cara kerja array dalam Go, termasuk cara menghitung rata-rata dan mencari nilai ekstrim (tertinggi dan terendah).

III. UNGUIDED

1.

```
package main

import (
    "fmt"
    "math"
)

func j(px, py, sx, sy float64) float64{
    return math.Sqrt(math.Pow(px-sx, 2) + math.Pow(py-sy,
2))
}

func dalam(cx, cy, r, x, y float64) bool {
    return j(cx, cy, x, y) <= r
}

func main() {
    var centralx1, centraly1, radius1 float64
    var centralx2, centraly2, radius2 float64
    var x, y float64
    fmt.Scan(&centralx1, centraly1, radius1)
    fmt.Scan(&centralx2, centraly2, radius2)
    fmt.Scan(&x, &y)
    dalam1 := dalam(centralx1, centraly1, radius1, x, y)
    dalam2 := dalam(centralx2, centraly2, radius2, x, y)
```

```

    if dalam1 && dalam2 {
        fmt.Print("Titik didalam lingkaran 1 dan 2")
    } else if dalam1{
        fmt.Print("Titik didalam lingkaran 1")
    } else if dalam2{
        fmt.Print("Titik didalam lingkaran 2")
    }
}

```

Output Screenshot:

```

● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
1 1 5
8 8 4
2 2
Titik di dalam lingkaran 1
● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
1 2 3
4 5 6
7 8
Titik di dalam lingkaran 2
● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
5 10 15
-15 4 20
0 0
Titik di dalam lingkaran 1 dan 2
● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
1 1 5
8 8 4
15 20
Titik di luar lingkaran 1 dan 2
○ PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

Penjelasan:

Program ini berfungsi untuk menentukan apakah suatu titik berada di dalam dua lingkaran yang berbeda. Program pertama-tama menerima input berupa koordinat pusat dan radius dari dua lingkaran, serta koordinat titik yang akan diperiksa. Fungsi j digunakan untuk menghitung jarak antara dua titik, yaitu pusat lingkaran dan titik yang diperiksa, dengan menggunakan rumus jarak Euclidean. Fungsi d kemudian memeriksa apakah titik tersebut berada dalam lingkaran dengan membandingkan jarak titik dari pusat lingkaran dengan radius lingkaran. Di dalam fungsi $main$, program memeriksa apakah titik berada di dalam lingkaran pertama, kedua, atau keduanya, dan menampilkan hasilnya. Program ini berguna untuk aplikasi yang melibatkan geometri dan pengujian kedekatan titik terhadap lingkaran.

rumus matematika yang berlaku. Jika rr lebih besar dari nn , hasil perhitungan dikembalikan sebagai 0.

1.

```
package main

import (
    "fmt"
    "math"

    func main() {
        var n int

        fmt.Print("Masukkan jumlah elemen array: ")
        fmt.Scan(&n)

        array := make([]int, n)

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

        // a. Menampilkan keseluruhan isi array
        fmt.Println("\na. Isi keseluruhan array:")
        fmt.Println(array)

        for i := 0; i < Len(array); i++ {
            fmt.Printf("array[%d] = %d\n", i, array[i])
        }

        // b. Menampilkan elemen dengan indeks ganjil
        fmt.Println("\nb. Elemen dengan indeks ganjil:")
        for i := 1; i < Len(array); i += 2 {
            fmt.Printf("array[%d] = %d\n", i, array[i])
        }
    }
}
```

```

// c. Menampilkan elemen dengan indeks genap
fmt.Println("\nc. Elemen dengan indeks genap:")
for i := 0; i < Len(array); i += 2 {
    fmt.Printf("array[%d] = %d\n", i, array[i])
}

// d. Cetak elemen kelipatan x
var x int
fmt.Print("\nd. Masukkan angka untuk kelipatan indeks: ")
fmt.Scan(&x)
fmt.Printf("Elemen dengan indeks kelipatan %d:\n", x)
if x != 0 {
    for i := 0; i < Len(array); i++ {
        if i%x == 0 {
            fmt.Printf("array[%d] = %d\n", i, array[i])
        }
    }
} else {
    fmt.Println("Tidak bisa mencari kelipatan 0.")
}

// e. Hapus elemen Array
var hapus int
fmt.Print("\ne. Masukkan indeks yang ingin dihapus: ")
fmt.Scan(&hapus)
if hapus >= 0 && hapus < Len(array) {
    array = append(array[:hapus], array[hapus+1:]...)
    fmt.Println("Array setelah penghapusan:")
    fmt.Println(array)
}

```

```

        for i := 0; i < Len(array); i++ {
            fmt.Printf("array[%d] = %d\n", i, array[i])
        }
    } else {
        fmt.Println("Indeks tidak valid.")
    }

    // f. Rata-rata array
    fmt.Println("\nf. Rata-rata elemen array:")
    if Len(array) > 0 {
        sum := 0
        for i := 0; i < Len(array); i++ {
            sum += array[i]
        }
        rataRata := float64(sum) / float64(Len(array))
        fmt.Printf("%.2f\n", rataRata)

        // g. Standar deviasi
        fmt.Println("\ng. Standar deviasi elemen array:")
        total := 0.0
        for i := 0; i < Len(array); i++ {
            total += math.Pow(float64(array[i]) - rataRata, 2)
        }
        variance := total / float64(Len(array))
        sd := math.Sqrt(variance)
        fmt.Printf("%.2f\n", sd)
    } else {
        fmt.Println("Array kosong, tidak bisa menghitung rata-rata dan standar deviasi.")
    }
}

```



```
// h. Frekuensi dari suatu bilangan

var cari int

fmt.Print("\nh. Masukkan bilangan yang ingin dicari
frekuensinya: ")

fmt.Scan(&cari)

frekuensi := 0

for i := 0; i < Len(array); i++ {
    if array[i] == cari {
        frekuensi++
    }
}

fmt.Printf("Frekuensi %d dalam array adalah: %d\n", cari,
frekuensi)

}
```

Output Screenshot:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPR02LUTFI> go run
Masukkan jumlah elemen array: 5
Masukkan elemen-elemen array:
Elemen ke-0: 1
Elemen ke-1: 3
Elemen ke-2: 5
Elemen ke-3: 7
Elemen ke-4: 9

a. Isi keseluruhan array:
[1 3 5 7 9]
array[0] = 1
array[1] = 3
array[2] = 5
array[3] = 7
array[4] = 9

b. Elemen dengan indeks ganjil:
array[1] = 3
array[3] = 7

c. Elemen dengan indeks genap:
array[0] = 1
array[2] = 5
array[4] = 9

d. Masukkan angka untuk kelipatan indeks: 3
Elemen dengan indeks kelipatan 3:
array[0] = 1
array[3] = 7

e. Masukkan indeks yang ingin dihapus: 2
Array setelah penghapusan:
[1 3 7 9]
array[0] = 1
array[1] = 3
array[2] = 7
array[3] = 9

f. Rata-rata elemen array:
5.00

g. Standar deviasi elemen array:
3.16

h. Masukkan bilangan yang ingin dicari frekuensinya: 9
Frekuensi 9 dalam array adalah: 1
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPR02LUTFI> █
```

Penjelasan:

Program ini memungkinkan pengguna untuk melakukan berbagai operasi pada array integer, termasuk memasukkan elemen array, serta menghitung dan menampilkan beberapa statistik terkait array. Program dimulai dengan meminta pengguna untuk memasukkan jumlah elemen array dan nilai-nilai setiap elemen tersebut. Selanjutnya, program menampilkan seluruh isi array, elemen dengan indeks ganjil, elemen dengan indeks genap, dan elemen dengan indeks kelipatan tertentu yang dipilih oleh pengguna. Program juga memungkinkan pengguna untuk menghapus elemen array berdasarkan indeks yang dimasukkan dan menampilkan hasil array setelah penghapusan. Selain itu, program menghitung dan menampilkan rata-rata elemen array, standar deviasi, serta frekuensi kemunculan bilangan tertentu dalam array. Jika array kosong, program akan memberikan peringatan saat mencoba menghitung rata-rata dan standar deviasi. Program ini sangat berguna untuk manipulasi dan analisis data array secara dinamis.

3.

```
package main

import "fmt"

func main() {

    var klubA, klubB string

    var poinA, poinB int

    var pemenang []string

    fmt.Print("Klub A: ")

    fmt.Scan(&klubA)

    fmt.Print("Klub B: ")

    fmt.Scan(&klubB)

    pertandingan :=

    for {

        fmt.Printf("Pertandingan %d (skor %s dan %s): ", pertandingan,
klubA, klubB)

        fmt.Scan(&poinA, &poinB)

        if poinA < 0 || poinB < 0 {

            break

        }

        if poinA > poinB {

            pemenang = append(pemenang, klubA)

        } else if poinB > poinA {

            pemenang = append(pemenang, klubB)

        } else {

            pemenang = append(pemenang, "Draw")

        }

        pertandingan++

    }

}
```

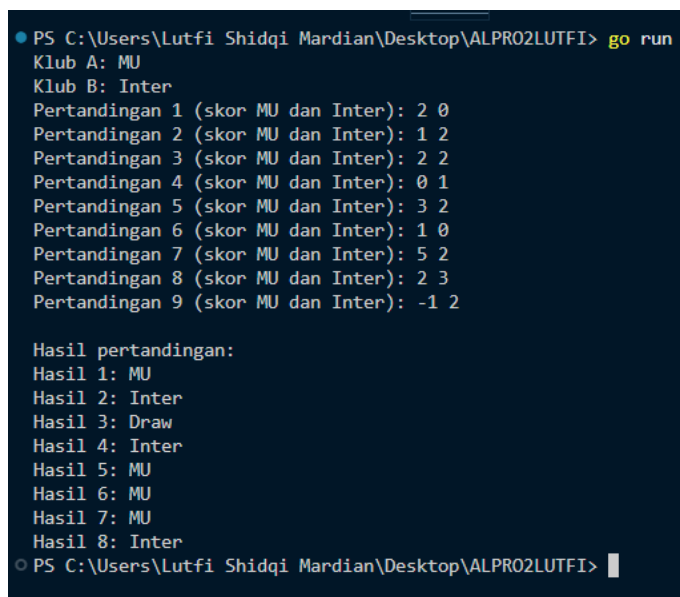
```

    fmt.Println("\nHasil pertandingan:")

    for i := 0; i < Len(pemenang); i++ {
        fmt.Printf("Hasil %d: %s\n", i+1, pemenang[i])
    }
}

```

Output Screenshot:



```

PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
Klub A: MU
Klub B: Inter
Pertandingan 1 (skor MU dan Inter): 2 0
Pertandingan 2 (skor MU dan Inter): 1 2
Pertandingan 3 (skor MU dan Inter): 2 2
Pertandingan 4 (skor MU dan Inter): 0 1
Pertandingan 5 (skor MU dan Inter): 3 2
Pertandingan 6 (skor MU dan Inter): 1 0
Pertandingan 7 (skor MU dan Inter): 5 2
Pertandingan 8 (skor MU dan Inter): 2 3
Pertandingan 9 (skor MU dan Inter): -1 2

Hasil pertandingan:
Hasil 1: MU
Hasil 2: Inter
Hasil 3: Draw
Hasil 4: Inter
Hasil 5: MU
Hasil 6: MU
Hasil 7: MU
Hasil 8: Inter
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

Penjelasan:

Program ini digunakan untuk mencatat dan menampilkan hasil pertandingan antara dua klub sepak bola. Pengguna diminta untuk memasukkan nama kedua klub dan kemudian memasukkan skor masing-masing klub untuk setiap pertandingan. Program akan terus menerima input skor pertandingan hingga skor yang dimasukkan adalah nilai negatif, yang akan menghentikan input lebih lanjut. Setelah itu, program mencatat pemenang setiap pertandingan (klub yang mencetak lebih banyak gol) atau mencatat hasil imbang jika skor kedua klub sama. Program kemudian menampilkan hasil dari semua pertandingan yang telah dimasukkan, dengan mencetak hasil masing-masing pertandingan, apakah kemenangan untuk salah satu klub atau hasil imbang.

4.

```
package main

import "fmt"

const NMAX int = 127

type tabel [NMAX]rune

func isiArray(t *tabel, n *int) {

    var input rune

    for {

        fmt.Scanf("%c", &input)

        if input == '.' // *n >= NMAX {

            break

        }

        t[*n] = input

        *n++

    }

}

func cetakArray(t tabel, n int) {

    for i := 0; i < n; i++ {

        fmt.Printf("%c ", t[i])

    }

    fmt.Println()

}
```

```
func balikanArray(t *tabel, n int) {
```

```
    for i := 0; i < n/2; i++ {
```

```
        t[i], t[n-1-i] = t[n-1-i], t[i]
```

```
    }
```

```
}
```

```
func palindrom(t tabel, n int) bool {
```

```
    for i := 0; i < n/2; i++ {
```

```
        if t[i] != t[n-1-i] {
```

```
            return false
```

```
        }
```

```
    }
```

```
    return true
```

```
}
```

```
func main() {
```

```
    var tab tabel
```

```
    var m int
```

```
    isiArray(&tab, &m)
```

```
    fmt.Println("\nTeks:")
```

```
    cetakArray(tab, m)
```

```
    balikanArray(&tab, m)
```

```
    fmt.Println("Reverse Teks:")
```

```
    cetakArray(tab, m)
```

```
if palindrom(tab, m) {  
    fmt.Println("Palindrom? true")  
} else {  
    fmt.Println("Palindrom? false")  
}  
}
```

Ss Output:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run  
K A T A K.  
  
Teks:  
K A T A K  
Reverse Teks:  
K A T A K  
Palindrom? true  
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run  
S E N A N G.  
  
Teks:  
S E N A N G  
Reverse Teks:  
G N A N E S  
Palindrom? false  
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> █
```

Penjelasan:

Program ini berfungsi untuk membaca rangkaian karakter dari input pengguna dan menyimpannya ke dalam sebuah array bertipe rune. Proses input berlangsung hingga pengguna memasukkan tanda titik (.) atau jumlah karakter mencapai batas maksimum yang telah ditentukan (NMAX = 127). Setelah input selesai, program mencetak seluruh karakter yang telah dimasukkan satu per satu dengan spasi sebagai pemisah. Kemudian,

program membalik urutan karakter dalam array menggunakan fungsi `balikanArray`, di mana elemen pertama ditukar dengan elemen terakhir, elemen kedua dengan elemen kedua terakhir, dan seterusnya. Hasil pembalikan ini kemudian dicetak ulang agar pengguna dapat melihat bentuk teks setelah dibalik.

Selanjutnya, program melakukan pemeriksaan apakah rangkaian karakter tersebut merupakan palindrom menggunakan fungsi `palindrom`. Sebuah teks disebut palindrom jika urutan karakter dari depan ke belakang sama dengan urutan dari belakang ke depan. Proses pengecekan dilakukan dengan membandingkan pasangan karakter dari ujung ke tengah. Jika semua pasangan cocok, program akan menampilkan hasil `Palindrom? true`; jika tidak, akan ditampilkan `Palindrom? false`. Dengan demikian, program ini tidak hanya mengelola input karakter dan pembalikannya, tetapi juga menguji simetri teks secara menyeluruh.

IV. KESIMPULAN

Dari berbagai implementasi tipe bentukan, array, slice, dan map dalam program yang telah dibuat, dapat disimpulkan bahwa penggunaan tipe bentukan seperti alias dan struct dalam pemrograman Golang sangat membantu dalam menyederhanakan representasi data yang kompleks serta meningkatkan keterbacaan program. Array dan slice memudahkan penyimpanan dan pengelolaan kumpulan data secara sistematis, sedangkan map memungkinkan pengelolaan data berbasis pasangan kunci dan nilai yang fleksibel. Pemahaman konsep seperti deklarasi tipe baru, manipulasi elemen array dan slice, serta penggunaan map untuk akses data dinamis sangat penting untuk membangun program yang modular, efisien, dan mudah dikembangkan. Dengan penerapan struktur data yang tepat, program menjadi lebih terorganisir, mudah diperbaiki, dan siap menghadapi kebutuhan pemrosesan data yang lebih kompleks.

V. REFERENSI

1. Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
2. Official Golang Documentation. (n.d.). Structs in Go. Retrieved from https://go.dev/doc/effective_go#structs
3. Official Golang Documentation. (n.d.). Arrays, Slices (and Strings): The Mechanics of 'append'. Retrieved from <https://go.dev/blog/slices-intro>
4. W3Schools. (n.d.). Golang Structs Tutorial. Retrieved from https://www.w3schools.com/go/go_structs.php
5. GeeksforGeeks. (n.d.). Arrays in Golang. Retrieved from <https://www.geeksforgeeks.org/arrays-in-golang/>
6. Ardan Labs. (n.d.). Map Internals in Go. Retrieved from <https://www.ardanlabs.com/blog/2013/12/using-maps-in-go.html>