LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 7

STRUCK & ARRAY

Oleh:

NAMA: ICHYA ULUMIDDIIN

NIM: 103112400076

KELAS: 12IF-01-A

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

a) Struct adalah tipe data komposit yang memungkinkan pengelompokan beberapa variabel dengan tipe data yang berbeda ke dalam satu entitas. Struct sering digunakan untuk merepresentasikan objek atau entitas dalam program.

   Karakteristik Struct:

   - Menyatukan beberapa field dengan tipe data yang berbeda.
   - Digunakan untuk merepresentasikan entitas kompleks.
   - Dapat diinisialisasi dan dimodifikasi sesuai kebutuhan.

b) Array adalah koleksi elemen dengan tipe data yang sama dan ukuran tetap. Array digunakan untuk menyimpan data dalam jumlah tetap yang diketahui sebelumnya.

   Karakteristik Array:

   - Ukuran tetap yang ditentukan saat deklarasi.
   - Setiap elemen diakses menggunakan indeks yang dimulai dari 0.
   - Digunakan ketika jumlah elemen sudah diketahui dan tidak berubah.

c) Slice adalah struktur data yang memberikan tampilan dinamis terhadap array. Berbeda dengan array, slice dapat berubah ukuran selama program berjalan.

   Karakteristik Slice:

   - Ukuran dinamis yang dapat berubah selama eksekusi program.
   - Dibangun di atas array, sehingga lebih fleksibel.
   - Dapat ditambahkan elemen baru menggunakan fungsi append.
   - Memiliki properti length dan capacity yang dapat diperiksa menggunakan fungsi len dan cap.

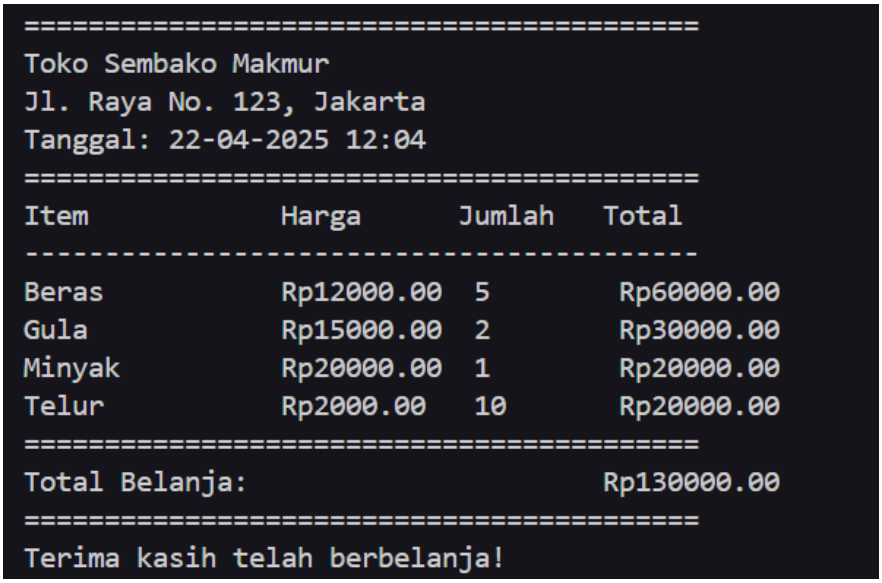
II. GUIDED

Source Code 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



Penjelasan

Program ini adalah simulasi struk belanja menggunakan bahasa pemrograman Go (Golang) yang memanfaatkan konsep struct dan method. Struct `Item` digunakan untuk merepresentasikan barang belanjaan yang memiliki atribut nama, harga, dan jumlah. Struct `Receipt` menyimpan informasi lengkap tentang transaksi, seperti nama toko, waktu belanja, daftar barang, serta total harga seluruh belanjaan.

Dalam program ini, method `CalculateTotal` digunakan untuk menghitung total harga belanja dengan menjumlahkan hasil perkalian harga dan jumlah tiap item. Sedangkan method `PrintReceipt` mencetak struk belanja dalam format tabel yang rapi, lengkap dengan waktu dan total pembayaran. Di fungsi `main`, dibuat sebuah objek `Receipt` dengan beberapa item contoh, lalu program menghitung total dan mencetak struk ke layar.

Tujuan utama program ini adalah menunjukkan bagaimana Go dapat digunakan untuk membuat sistem pencatatan transaksi sederhana yang efisien dan terstruktur.

Source Code Guided 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

```
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
```

Penjelasan

Program ini bertujuan untuk memperlihatkan cara penggunaan array satu dimensi dan dua dimensi dalam bahasa Go untuk mengelola data nilai mahasiswa. Pada bagian pertama, array `nilaiMahasiswa` menyimpan lima nilai yang kemudian ditampilkan, dihitung rata-ratanya, serta dicari nilai tertinggi dan terendahnya.

Di bagian kedua, digunakan array dua dimensi `nilaiUjian` untuk menyimpan nilai dua mata kuliah (Matematika dan Bahasa) bagi tiga mahasiswa, lalu ditampilkan dengan format yang terstruktur. Program ini memberikan gambaran dasar tentang pengolahan data numerik menggunakan array di Go, termasuk perhitungan statistik sederhana dan pengelolaan data multidimensi.

III. UNGUIDED

Source Code Unguided 1

```
//ICHYA ULUMIDDIIN
package main

import (
    "fmt"
    "math"
)
func jarak(a, b, c, d float64) float64 {
    return math.Sqrt(math.Pow(a-c, 2) + math.Pow(b-d,
2))
}
func didalam(cx, cy, r, x, y float64) bool {
    return jarak(cx, cy, x, y) <= r
}
func main() {
    var cx1, cy1, r1 float64
    var cx2, cy2, r2 float64
    var x, y float64
    fmt.Scan(&cx1, &cy1, &r1)
    fmt.Scan(&cx2, &cy2, &r2)
    fmt.Scan(&x, &y)
    dalamLingkaran1 := didalam(cx1, cy1, r1, x, y)
    dalamLingkaran2 := didalam(cx2, cy2, r2, x, y)
    if dalamLingkaran1 && dalamLingkaran2 {
        fmt.Println("Titik di dalam lingkaran 1 dan 2")
    } else if dalamLingkaran1 {
        fmt.Println("Titik di dalam lingkaran 1")
    } else if dalamLingkaran2 {
        fmt.Println("Titik di dalam lingkaran 2")
    } else {
        fmt.Println("Titik di luar lingkaran 1 dan 2")
    }
}
```

Output

```
PS C:\Users\ICHYA ULUMIDDIN\OneDrive\
uided1\Unguided1.go"
1 1 5
8 8 4
2 2
Titik di dalam lingkaran 1
```

Penjelasan

Program terdiri dari dua fungsi utama. Fungsi `jarak` menghitung jarak antara dua titik berdasarkan koordinatnya, sementara fungsi `didalam` menggunakan hasil perhitungan jarak untuk menentukan apakah suatu titik berada di dalam sebuah lingkaran dengan membandingkan jarak titik ke pusat lingkaran dengan jari-jari lingkaran tersebut.

Pada bagian utama (`main`), pengguna diminta untuk memasukkan data dua lingkaran (koordinat pusat dan jari-jari) serta satu titik yang akan diperiksa. Program kemudian memanggil fungsi `didalam` untuk kedua lingkaran dan menentukan posisi titik tersebut berdasarkan hasil pengecekan. Hasilnya ditampilkan ke layar dalam bentuk pesan teks yang menjelaskan apakah titik berada di dalam lingkaran pertama, kedua, keduanya, atau di luar keduanya.

Source Code Unguided 2

```
package main

import (
	"fmt"
	"math"
)
func main() {
	var n int
	fmt.Print("Masukkan jumlah elemen array: ")
	fmt.Scan(&n)
	array := make([]int, n)
	for i := 0; i < n; i++ {
		fmt.Printf("Masukkan elemen ke-%d: ", i)
		fmt.Scan(&array[i])
	}
	fmt.Println("\na. Menampilkan semua elemen array:")
	fmt.Println(array)
	// b. Menampilkan elemen dengan indeks ganjil
	fmt.Println("b. Elemen dengan indeks ganjil:")
	for i := 1; i < len(array); i += 2 {
		fmt.Printf("Indeks %d: %d\n", i, array[i])
	}
	// c. Menampilkan elemen dengan indeks genap
	fmt.Println("c. Elemen dengan indeks genap:")
	for i := 0; i < len(array); i += 2 {
		fmt.Printf("Indeks %d: %d\n", i, array[i])
	}
	// d. Menampilkan elemen dengan indeks kelipatan x
	var x int
	fmt.Print("d. Masukkan nilai x (untuk mencari indeks
kelipatan x): ")
	fmt.Scan(&x)
	fmt.Printf("   Elemen dengan indeks kelipatan
%d:\n", x)
	for i := 0; i < len(array); i++ {
		if i%x == 0 {
			fmt.Printf("Indeks %d: %d\n", i,
array[i])
		}
	}
```

```
// e. Menghapus elemen pada indeks tertentu
	var indeksHapus int
	fmt.Print("e. Masukkan indeks yang ingin dihapus: ")
	fmt.Scan(&indeksHapus)
	if indeksHapus >= 0 && indeksHapus < len(array) {
		array = append(array[:indeksHapus], array[indeksHapus+1:]...)
		fmt.Println("   Isi array setelah elemen dihapus:")
		fmt.Println(array)
	} else {
		fmt.Println("   Indeks tidak valid!")
	}
	// f. Menampilkan rata-rata
	total := 0
	for _, v := range array {
		total += v
	}
	rataRata := float64(total) / float64(len(array))
	fmt.Printf("f. Rata-rata dari elemen array: %.2f\n", rataRata)
	// g. Menampilkan simpangan baku (standar deviasi)
	var jumlahSelisihKuadrat float64
	for _, v := range array {
		selisih := float64(v) - rataRata
		jumlahSelisihKuadrat += selisih * selisih
	}
	simpanganBaku := math.Sqrt(jumlahSelisihKuadrat / float64(len(array)))
	fmt.Printf("g. Simpangan baku dari elemen array: %.2f\n", simpanganBaku)
	// h. Menampilkan frekuensi kemunculan bilangan tertentu
	var bilanganCari int
	fmt.Print("h. Masukkan bilangan yang ingin dicari frekuensinya: ")
	fmt.Scan(&bilanganCari)
	frekuensi := 0
	for _, v := range array {
		if v == bilanganCari {
			frekuensi++
		}
	}
	fmt.Printf("   Frekuensi kemunculan angka %d: %d kali\n", bilanganCari, frekuensi)
}
```

Output

```
Masukkan jumlah elemen array: 5
Masukkan elemen ke-0: 3
Masukkan elemen ke-1: 1
Masukkan elemen ke-2: 2
Masukkan elemen ke-3: 4
Masukkan elemen ke-4: 5

a. Menampilkan semua elemen array:
[3 1 2 4 5]
b. Elemen dengan indeks ganjil:
Indeks 1: 1
Indeks 3: 4
c. Elemen dengan indeks genap:
Indeks 0: 3
Indeks 2: 2
Indeks 4: 5
d. Masukkan nilai x (untuk mencari indeks kelipatan x): 9
   Elemen dengan indeks kelipatan 9:
Indeks 0: 3
e. Masukkan indeks yang ingin dihapus: 1
   Isi array setelah elemen dihapus:
[3 2 4 5]
f. Rata-rata dari elemen array: 3.50
g. Simpangan baku dari elemen array: 1.12
h. Masukkan bilangan yang ingin dicari frekuensinya: 20
   Frekuensi kemunculan angka 20: 0 kali
```

Penjelasan

Program dimulai dengan meminta pengguna menentukan jumlah elemen array, kemudian menerima input nilai-nilai tersebut. Setelah array terbentuk, program menampilkan semua elemen array (a), lalu menampilkan elemen dengan indeks ganjil (b), genap (c), dan indeks yang merupakan kelipatan dari suatu angka yang dimasukkan pengguna (d). Selanjutnya, program memberikan fitur penghapusan elemen berdasarkan indeks tertentu (e), yang menunjukkan penggunaan fungsi append untuk memodifikasi slice (array dinamis di Go). Kemudian, program menghitung rata-rata dari seluruh elemen array (f), dan dilanjutkan dengan perhitungan simpangan baku (standar deviasi) (g), yang merupakan ukuran statistik untuk mengetahui seberapa jauh data tersebar dari rata-ratanya.

Akhirnya, program juga memungkinkan pengguna mencari frekuensi kemunculan suatu bilangan dalam array (h), yaitu menghitung seberapa sering bilangan tertentu muncul.

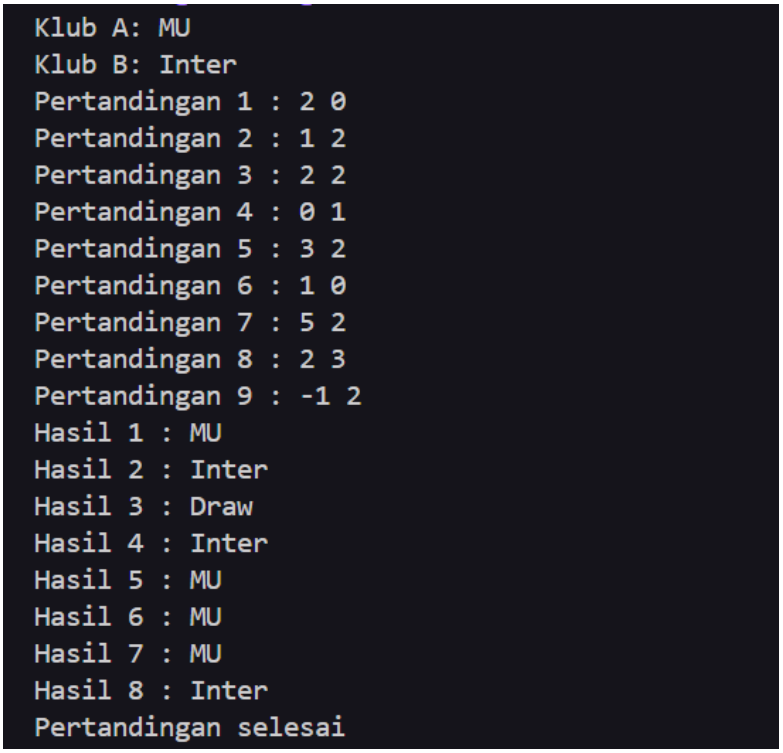
Source Code Unguided 3

```
package main

import (
	"fmt"
)

func main() {
	var klubA, klubB string
	var skorA, skorB int
	var hasil []string
	fmt.Print("Klub A: ")
	fmt.Scanln(&klubA)
	fmt.Print("Klub B: ")
	fmt.Scanln(&klubB)
	pertandingan := 1
	for {
		fmt.Printf("Pertandingan %d : ",
pertandingan)
		fmt.Scan(&skorA, &skorB)
		if skorA < 0 || skorB < 0 {
			break
		}
		if skorA > skorB {
			hasil = append(hasil, klubA)
		} else if skorB > skorA {
			hasil = append(hasil, klubB)
		} else {
			hasil = append(hasil, "Draw")
		}
		pertandingan++
	}
	for i, v := range hasil {
		fmt.Printf("Hasil %d : %s\n", i+1, v)
	}
	fmt.Println("Pertandingan selesai")
}
```

Output



Penjelasan

Program di atas bertujuan untuk mencatat dan menampilkan hasil dari serangkaian pertandingan antara dua klub sepak bola. Setiap kali skor dimasukkan, program membandingkan nilai skor klub A dan klub B. Jika skor klub A lebih tinggi, maka nama klub A dicatat sebagai pemenang pertandingan tersebut; sebaliknya, jika skor klub B lebih tinggi, maka klub B dicatat sebagai pemenang. Jika skor sama, maka hasil pertandingan dicatat sebagai "Draw". Semua hasil pertandingan disimpan dalam slice hasil. Setelah penginputan dihentikan, program akan mencetak seluruh hasil pertandingan berdasarkan urutan pertandingan yang telah berlangsung.

Source Code Unguided 4

```
package main

import (
	"fmt"
)
const NMAX int = 127
type tabel [NMAX]rune
func isiArray(t *tabel, n *int) {
	var ch rune
	*n = 0
	fmt.Print("Teks : ")
	for {
		fmt.Scanf("%c", &ch)
		// Hentikan kalau titik
		if ch == '.' || *n >= NMAX {
			break
		}
		// Abaikan spasi dan newline
		if ch != ' ' && ch != '\n' {
			(*t)[*n] = ch
			*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 {
	var tCopy tabel
	copy(tCopy[:], t[:])
	balikanArray(&tCopy, n)
	for i := 0; i < n; i++ {
		if t[i] != tCopy[i] {
			return false
		}
	}
	return true
}
```

```
func main() {
	var tab tabel
	var m int

	isiArray(&tab, &m)

	fmt.Print("Teks : ")
	cetakArray(tab, m)

	fmt.Printf("Palindrom ? %v\n", palindrom(tab, m))
}
```

Output

Sebelum di modifikasi:

```
PS C:\Users\ICHYA ULUMIDDIN\OneDrive\Documents\Alpro 2>
go run "c:\Users\ICHYA ULUMIDDIN\OneDrive\Documents\Alpro 2\Praktikum_week4\Unguided4.go"
Teks : S E N A N G.
Reverse teks : G N A N E S
```

Setelah di modifikasi:

```
PS C:\Users\ICHYA ULUMIDDIN\On
uided4\Unguided4.go"
Teks : K A T A K .
Teks : K A T A K
Palindrom ? true
PS C:\Users\ICHYA ULUMIDDIN\On
uided4\Unguided4.go"
Teks : S E N A N G .
Teks : S E N A N G
Palindrom ? false
```

Penjelasan

Program di atas merupakan implementasi dalam bahasa Go (Golang) yang bertujuan untuk membaca sekumpulan karakter dari input pengguna, menampilkannya, dan memeriksa apakah karakter-karakter tersebut membentuk kata palindrom. Program menggunakan sebuah array bertipe rune untuk menampung karakter, dan membatasi jumlah maksimal karakter sebanyak 127 melalui konstanta NMAX. Fungsi isiArray digunakan untuk membaca input karakter satu per satu dari pengguna hingga karakter titik (.) dimasukkan. Spasi dan newline akan diabaikan

agar hanya karakter penting yang diproses. Fungsi cetakArray akan menampilkan isi array ke layar, dengan setiap karakter dipisahkan oleh spasi.

Fungsi balikanArray berfungsi membalikkan isi array secara in-place (dalam tempat yang sama), sedangkan fungsi palindrom membuat salinan array asli, membalik salinan tersebut, dan membandingkannya dengan array asli. Jika seluruh karakter sama, maka fungsi ini mengembalikan true yang berarti input adalah palindrom.

Pada bagian main, program meminta pengguna untuk memasukkan teks, kemudian mencetak kembali teks tersebut, dan mengevaluasi apakah teks tersebut adalah palindrom atau tidak, lalu menampilkan hasilnya.

IV. KESIMPULAN

Array adalah struktur data yang menyimpan sekumpulan elemen dengan tipe data yang sama dalam satu variabel, dengan ukuran yang tetap. Dalam pemrograman Go, array memudahkan pengelolaan data yang memiliki tipe serupa, seperti daftar nilai atau karakter. Array dapat diakses menggunakan indeks yang dimulai dari 0, memungkinkan operasi seperti penambahan, pengubahan, dan pengecekan elemen secara efisien melalui perulangan. Dalam contoh program, array digunakan untuk menyimpan karakter-karakter dari input pengguna dan melakukan manipulasi seperti membalikkan urutan elemen dan memeriksa apakah teks tersebut merupakan palindrom.

Struct, di sisi lain, adalah tipe data komposit yang memungkinkan pengelompokan variabel dengan tipe data yang berbeda dalam satu entitas. Ini berguna untuk merepresentasikan objek atau entitas yang lebih kompleks. Struct membantu menyimpan data yang terorganisir dalam satu unit yang mudah dikelola. Dalam program yang dibahas, struct digunakan untuk mengelompokkan data teks dan operasi terkait secara terstruktur. Kombinasi penggunaan array dan struct memungkinkan pengelolaan data lebih kompleks, seperti dalam pengecekan palindrom, di mana array menyimpan karakter dan struct mengatur operasi terkait yang memudahkan pengolahan data.

.

REFERENSI

Cheney, D. (2018, July 12). *Slices from the ground up*. Retrieved from dave.cheney: https://dave.cheney.net/2018/07/12/slices-from-the-ground-up

Go. (2024, November 24). *Slices of Structs: Storing Complex Data in Go.* Retrieved from Sling Academy: https://www.slingacademy.com/article/go-slices-of-structs-storing-complex-data/

Parker, N. (2022, March 4). *Understanding Type Aliases in Go: A Comprehensive Guide*. Retrieved from Coding Explorations: https://www.codingexplorations.com/blog/understanding-type-aliases-in-go-a-comprehensive-guide