LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL VII

STRUCT & ARRAY

Oleh:

Muhammad Rifki Fadhilah

2311102032

IF 11 02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024/2025

I. DASAR TEORI

Array

Array adalah struktur data yang digunakan untuk menyimpan sekumpulan elemen dengan tipe data yang sama dalam satu variabel. Elemen-elemen dalam array diakses menggunakan indeks, dengan indeks pertama dimulai dari 0.

Contoh dasar:

1. Array bisa menyimpan angka, string, atau tipe data lainnya dalam jumlah tetap.
2. Ukuran array ditentukan pada saat deklarasi dan tidak bisa berubah setelah itu.

Struct

Struct adalah tipe data yang memungkinkan kita untuk menyimpan beberapa nilai dengan tipe yang berbeda dalam satu entitas. Dengan menggunakan struct, kita bisa mengelompokkan data yang berbeda (misalnya, nama, umur, tinggi) dalam satu objek.

Contoh dasar:

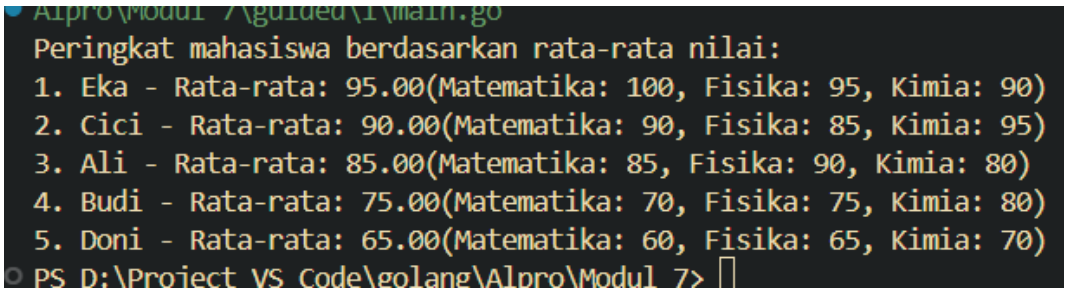
1. Dalam struct, kita mendefinisikan field dengan nama dan tipe data yang berbeda.
2. Struct sering digunakan untuk merepresentasikan objek dunia nyata, seperti mahasiswa, mobil, atau produk.

II. GUIDED

1. SOURCE CODE

```
package main

import(
  "fmt"
  "sort"
)

type Mahasiswa struct{
  Nama string
  Matematika int
  Fisika int
  Kimia int
  RataRata float64
}

func hitungRataRata(m *Mahasiswa){
  total:= m.Matematika + m.Fisika  + m.Kimia
  m.RataRata = float64(total) / 3.0
}

func main(){
  mahasiswa := []Mahasiswa{
    {"Ali",85,90,80,0},
    {"Budi",70,75,80,0},
    {"Cici",90,85,95,0},
    {"Doni",60,65,70,0},
    {"Eka",100,95,90,0},
  }
  for i := range mahasiswa {
    hitungRataRata(&mahasiswa[i])
  }

  sort.Slice(mahasiswa,func (i,j int)bool  {
    return mahasiswa[i].RataRata > mahasiswa[j].RataRata
  })

  fmt.Println("Peringkat mahasiswa berdasarkan rata-rata nilai:")
  for i,m := range mahasiswa{
    fmt.Printf("%d. %s - Rata-rata: %.2f(Matematika: %d, Fisika:
%d, Kimia: %d)\n",
      i+1,m.Nama,m.RataRata,m.Matematika,m.Fisika,m.Kimia)
  }
}
```

OUTPUT



DESKRIPSI PROGRAM

Program ini dibuat untuk menghitung rata-rata nilai tiga mata pelajaran (Matematika, Fisika, dan Kimia) bagi beberapa mahasiswa, kemudian mengurutkan mereka berdasarkan nilai rata-rata tersebut dari yang tertinggi ke terendah. Pertama, program ini menggunakan struct bernama Mahasiswa untuk menyimpan data penting setiap mahasiswa, termasuk nama, nilai pada masing-masing mata pelajaran, dan nilai rata-rata yang akan dihitung kemudian.

Dalam fungsi utama (main), program memulai dengan mendefinisikan sebuah slice yang berisi data lima mahasiswa. Setiap mahasiswa diberi nilai pada Matematika, Fisika, dan Kimia, sementara nilai rata-rata diinisialisasi ke nol. Selanjutnya, program melakukan iterasi pada slice tersebut dan menghitung rata-rata masing-masing mahasiswa dengan memanggil fungsi hitungRataRata. Fungsi ini menerima pointer ke data mahasiswa, menghitung total nilai ketiga mata pelajaran, dan mengonversinya menjadi rata-rata dengan membaginya dengan tiga. Nilai rata-rata tersebut kemudian disimpan di dalam struct Mahasiswa.

Setelah semua mahasiswa memiliki nilai rata-rata, program mengurutkan slice mahasiswa menggunakan fungsi sort.Slice. Pada fungsi pengurutan ini, setiap mahasiswa dibandingkan berdasarkan nilai rata-rata, sehingga mahasiswa dengan nilai rata-rata lebih tinggi berada di posisi lebih awal dalam urutan slice. Dengan demikian, mahasiswa diurutkan dari nilai rata-rata tertinggi ke terendah.

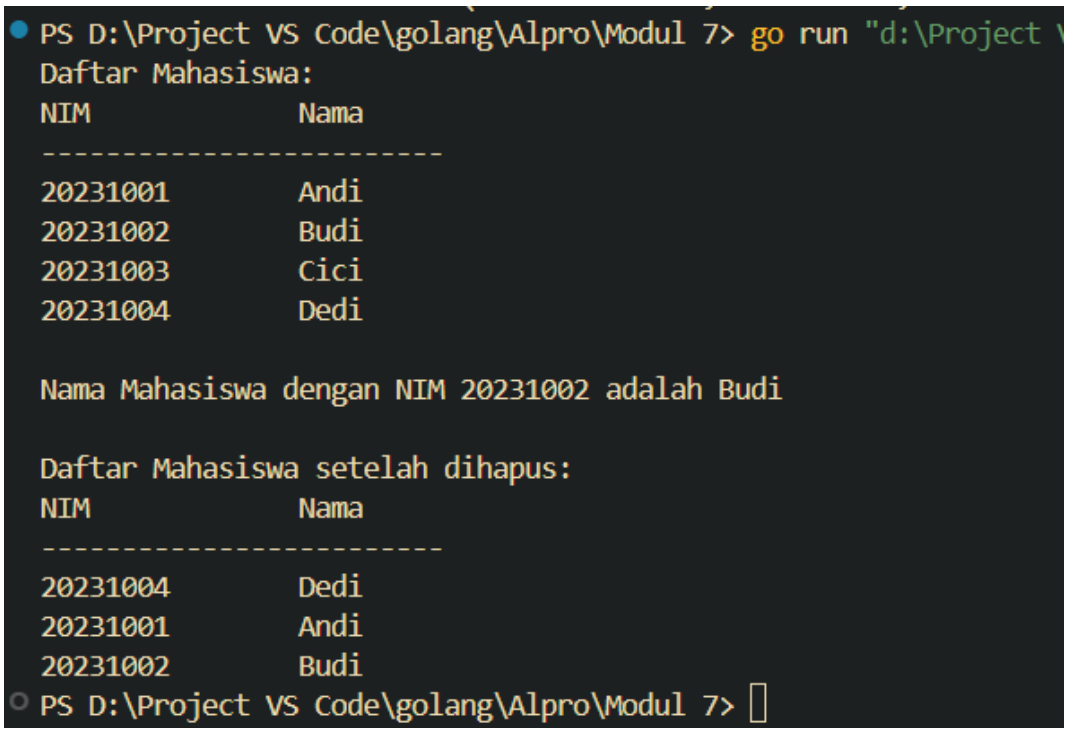
Akhirnya, program mencetak hasil urutan mahasiswa berdasarkan rata-rata nilai mereka. Untuk setiap mahasiswa, nama, rata-rata,

serta nilai masing-masing mata pelajaran ditampilkan. Program ini menunjukkan cara mengelola data kompleks menggunakan struct, pointer, serta cara mengurutkan data pada slice.

2. SOURCE CODE

```go
package main

import "fmt"

func main() {
  // Membuat map dengan NIM sebagai kunci dan Nama sebagai nilai
  mahasiswa := map[string]string{
    "20231001": "Andi",
    "20231002": "Budi",
    "20231003": "Cici",
  }

  // Menambahkan data baru ke map
  mahasiswa["20231004"] = "Dedi"

  // Menampilkan seluruh isi map dalam format kolom dan baris
  fmt.Println("Daftar Mahasiswa:")
  fmt.Println("NIM\t\tNama")
  fmt.Println("-------------------------")
  for nim, nama := range mahasiswa {
    fmt.Printf("%s\t%s\n", nim, nama)
  }

  // Mengakses data berdasarkan NIM
  nim := "20231002"
  fmt.Println("\nNama Mahasiswa dengan NIM", nim, "adalah",
mahasiswa[nim])

  // Menghapus data berdasarkan NIM
  delete(mahasiswa, "20231003")

  // Menampilkan isi map setelah data dihapus dalam format kolom
dan baris
  fmt.Println("\nDaftar Mahasiswa setelah dihapus:")
  fmt.Println("NIM\t\tNama")
  fmt.Println("-------------------------")
  for nim, nama := range mahasiswa {
    fmt.Printf("%s\t%s\n", nim, nama)
  }

}
```

OUTPUT



DESKRIPSI PROGRAM

Program ini menggunakan struktur data map dalam bahasa Go untuk menyimpan dan mengelola daftar mahasiswa dengan Nomor Induk Mahasiswa (NIM) sebagai kunci dan nama mahasiswa sebagai nilainya. Pertama, program membuat map bernama `mahasiswa`, di mana setiap elemen terdiri dari pasangan NIM sebagai kunci (tipe data `string`) dan nama sebagai nilai (tipe data `string`). Awalnya, map ini diisi dengan tiga mahasiswa: Andi, Budi, dan Cici, dengan masing-masing NIM unik.

Di bagian selanjutnya, data mahasiswa baru ditambahkan ke map menggunakan `mahasiswa["20231004"] = "Dedi"`. Data ini menambahkan mahasiswa baru bernama Dedi dengan NIM `20231004` ke dalam map. Setelah itu, program mencetak seluruh isi map dalam format kolom dan baris dengan menggunakan perulangan `for`, di mana setiap pasangan `nim, nama` diakses dan dicetak.

Kemudian, program mengakses dan menampilkan data mahasiswa berdasarkan NIM. Pada contoh ini, program mengambil nama mahasiswa dengan NIM `20231002`, yang disimpan dalam

variabel `nim` dan mencetak nama mahasiswa tersebut (yaitu, Budi).

Setelah menampilkan nama berdasarkan NIM, program menghapus data mahasiswa dengan NIM `20231003` menggunakan `delete(mahasiswa, "20231003")`, yang menghilangkan pasangan NIM-nama yang bersangkutan (dalam hal ini, data Cici) dari map.

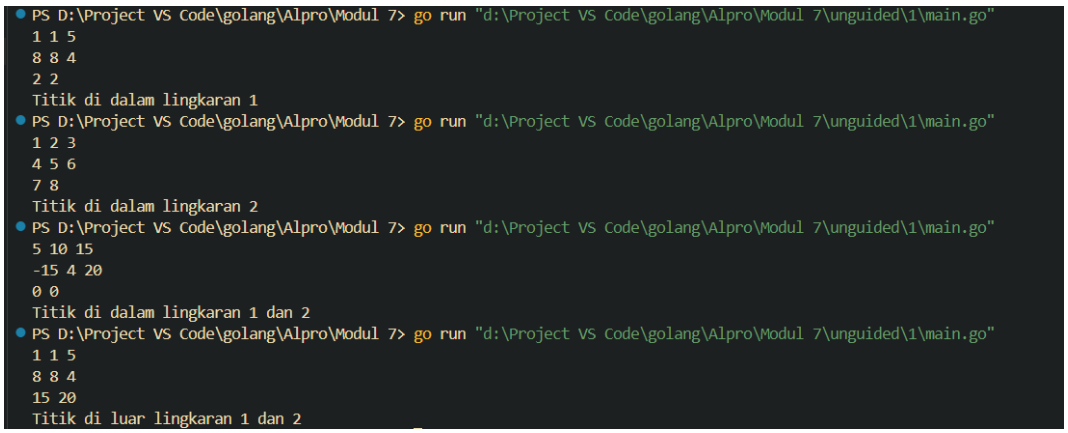
Akhirnya, program menampilkan kembali isi map setelah penghapusan data. Isi map ditampilkan dalam format yang sama seperti sebelumnya. Program ini menunjukkan penggunaan map untuk penyimpanan data yang efisien, pemanfaatan perulangan untuk mencetak elemen, serta cara menambah, mengakses, dan menghapus elemen dalam map di Go.

III. UNGUIDED

1. SOURCE CODE

```go
package main

import (
  "fmt"
  "math"
)

type titik int
type lingkaran int

func jarak(a, b, c, d titik) float64 {
  return math.Sqrt(math.Pow(float64(a-c), 2) +
math.Pow(float64(b-d), 2))
}

func didalam(cx, cy lingkaran, r lingkaran, x, y titik) bool {
  jarak_titik := jarak(x, y, titik(cx), titik(cy))
  return jarak_titik < float64(r)
}

func main() {
  var cx1, cy1, r1, cx2, cy2, r2, x, y titik

  fmt.Scanln(&cx1, &cy1, &r1)
  fmt.Scanln(&cx2, &cy2, &r2)
  fmt.Scanln(&x, &y)

  result1 := didalam(lingkaran(cx1), lingkaran(cy1),
lingkaran(r1), x, y)
  result2 := didalam(lingkaran(cx2), lingkaran(cy2),
lingkaran(r2), x, y)

  if result1 && result2 {
    fmt.Println("Titik di dalam lingkaran 1 dan 2")
  } else if result1 {
    fmt.Println("Titik di dalam lingkaran 1")
  } else if result2 {
    fmt.Println("Titik di dalam lingkaran 2")
  } else {
    fmt.Println("Titik di luar lingkaran 1 dan 2")
  }
}
```

OUTPUT



DESKRIPSI PROGRAM

Program ini bertujuan untuk menentukan apakah sebuah titik berada di dalam dua lingkaran tertentu, hanya di salah satu dari kedua lingkaran, atau di luar keduanya. Program ini menggunakan tipe data titik dan lingkaran, keduanya dideklarasikan sebagai alias tipe int untuk membedakan antara nilai koordinat titik dan radius lingkaran dalam program.

Fungsi jarak digunakan untuk menghitung jarak antara dua titik dalam bidang kartesius, dengan menggunakan rumus jarak Euclidean:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Fungsi ini menerima empat parameter bertipe titik, yaitu a, b, c, dan d, yang mewakili koordinat kedua titik. Jarak ini dikonversi ke tipe float64 untuk memastikan ketepatan hasil.

Fungsi didalam digunakan untuk memeriksa apakah titik (x, y) berada di dalam lingkaran dengan pusat (cx, cy) dan radius r. Fungsi ini memanfaatkan fungsi jarak untuk menghitung jarak antara titik (x, y) dan pusat lingkaran (cx, cy), kemudian membandingkan hasil jarak tersebut dengan radius lingkaran r. Jika jarak lebih kecil dari radius, maka titik berada di dalam lingkaran.

Dalam fungsi main, pertama-tama variabel-variabel yang diperlukan untuk mewakili pusat dan radius dua lingkaran, serta koordinat titik, dideklarasikan dengan tipe titik. Program meminta masukan dari pengguna untuk mengisi koordinat pusat dan radius kedua lingkaran, serta titik yang akan dicek posisinya.

Setelah itu, program memanggil fungsi didalam untuk masing-masing lingkaran dan menyimpan hasilnya dalam variabel result1 dan result2. Berdasarkan hasil ini, program menggunakan struktur kondisional if-else untuk menentukan apakah titik berada di dalam kedua lingkaran, hanya di dalam salah satu lingkaran, atau di luar keduanya. Output dari program berupa informasi posisi titik tersebut relatif terhadap kedua lingkaran.

2. SOURCE CODE

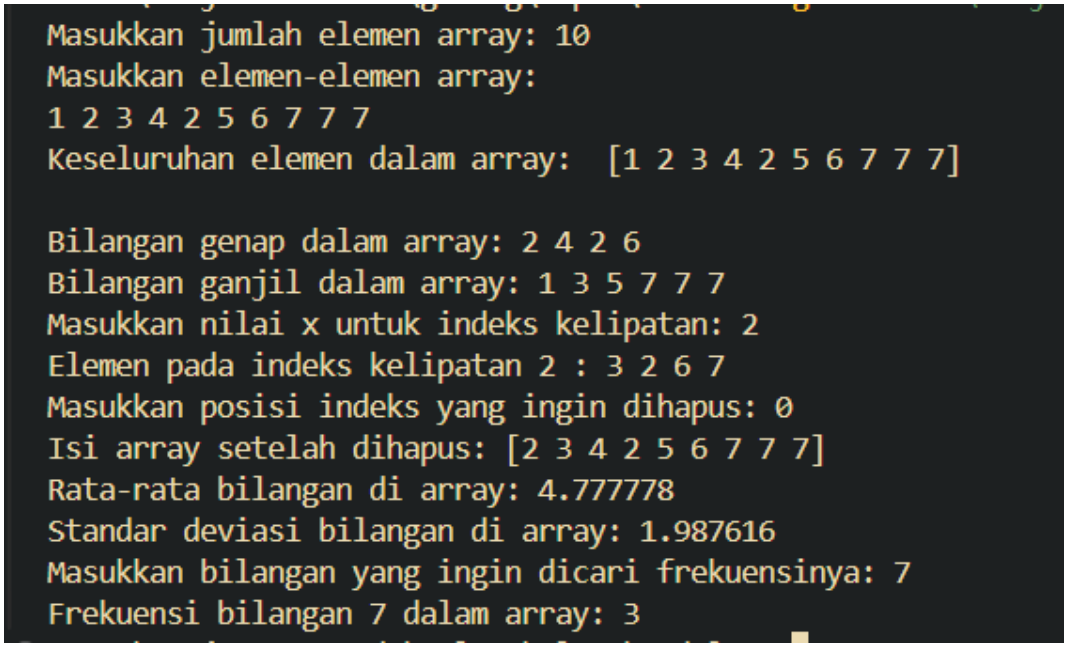
```go
package main

import(
  "fmt"
  "math"
)

func remove(array []int, s int) []int{
  return append(array[:s], array[s+1:]...)
}
```

```go
func average(arr []int, n int)float64{
  sum := 0
  for i := 0; i < n ; i++ {
    sum += (arr[i])
  }
  var avg float64 = float64(sum)/float64(n)
  return avg
}

func deviasi(arr []int) float64 {
  var sum, mean, sd float64
  n := float64(len(arr))

  // Konversi setiap elemen dari int ke float64 untuk perhitungan deviasi
  floatArr := make([]float64, len(arr))
  for i, value := range arr {
    floatArr[i] = float64(value)
    sum += floatArr[i]
  }

  mean = sum / n

  for _, value := range floatArr {
    sd += math.Pow(value-mean, 2)
  }
```

```
  sd = math.Sqrt(sd / n)
  return sd
}

func frekuensi(arr []int, bil int) int {
  count := 0
  for _, item := range arr {
    if item == bil {
      count++
    }
  }
  return count
}

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

  arr := make([]int, n)
  fmt.Println("Masukkan elemen-elemen array: ")
  for i := 0; i < n; i++ {
    fmt.Scan(&arr[i])
  }

  //Menampilkan keseluruhan isi array
  fmt.Println("Keseluruhan elemen dalam array: ", arr)
  fmt.Println()

  //Menampilkan bilangan genap dalam array
  fmt.Print("Bilangan genap dalam array: ")
  for i := 0; i < n; i++ {
    if arr[i]%2 == 0 {
      fmt.Print(arr[i]," ")
    }
  }
  fmt.Println()

  //Menampilkan bilangan ganjil dalam array
  fmt.Print("Bilangan ganjil dalam array: ")
  for i := 0; i < n; i++ {
    if arr[i]%2 == 1 {
      fmt.Print(arr[i]," ")
    }
  }
  fmt.Println()

  //Menampilkan bilangan dengan indeks kelipatan X
  var x int
  fmt.Print("Masukkan nilai x untuk indeks kelipatan: ")
```

```
  fmt.Scan(&x)

  if x <= 0 && x >= n {
    fmt.Print("Nilai x tidak valid")
  }

  fmt.Print("Elemen pada indeks kelipatan ", x, " : ")
  for i := x; i < n; i+= x {
    fmt.Print(arr[i]," ")
  }
  fmt.Println()

  //Menghapus elemen array dengan indeks tertentu dan menampilkan
keseluruhannya
  var index int
  fmt.Print("Masukkan posisi indeks yang ingin dihapus: ")
  fmt.Scan(&index)

  if index >= 0 && index < n{
    arr = remove(arr,index)
    fmt.Print("Isi array setelah dihapus: ", arr)
  }else{
    fmt.Print("Indeks tidak valid")
  }
  fmt.Println()

  fmt.Printf("Rata-rata bilangan di array: %2f\n",
average(arr,len(arr)))

  //Menampilkan standar deviasi dari bilangan dalam array
  fmt.Printf("Standar deviasi bilangan di array: %2f\n",
deviasi(arr))

  //Menampilkan frekuensi dari suatu bilangan dalam array
  var bil int
  fmt.Print("Masukkan bilangan yang ingin dicari frekuensinya: ")
  fmt.Scan(&bil)
  fmt.Printf("Frekuensi bilangan %d dalam array: %d\n", bil,
frekuensi(arr, bil))
}
```

OUTPUT



DESKRIPSI PROGRAM

Program ini ditulis dalam bahasa Go dan menyediakan berbagai fungsi untuk mengolah data dalam array. Pertama-tama, program menerima masukan dari pengguna berupa jumlah elemen array dan nilai dari setiap elemen tersebut. Setelah data dimasukkan, program menyediakan beberapa opsi analisis terhadap data yang tersimpan dalam array. Opsi pertama adalah menampilkan semua elemen dalam array, yang dilakukan langsung setelah array diisi.

Selanjutnya, program memberikan opsi untuk menampilkan elemen genap dan ganjil yang ada di dalam array. Hal ini dilakukan dengan melakukan pemeriksaan sederhana pada setiap elemen, di mana elemen yang habis dibagi dua ditampilkan sebagai elemen genap, sedangkan sisanya dikategorikan sebagai elemen ganjil. Program kemudian meminta masukan tambahan dari pengguna berupa nilai untuk menentukan indeks kelipatan yang akan ditampilkan. Misalnya, jika pengguna memasukkan nilai 2, program akan menampilkan elemen pada indeks kelipatan dua dalam array.

Pengguna juga memiliki opsi untuk menghapus elemen tertentu dari array dengan memasukkan indeks elemen yang ingin dihapus.

Setelah elemen tersebut dihapus, program menampilkan kembali array tanpa elemen yang telah dihapus.

Selain itu, program menghitung rata-rata dari semua elemen dalam array. Proses ini dilakukan dengan menambahkan semua elemen dan membaginya dengan jumlah elemen dalam array. Program juga dapat menghitung standar deviasi dari elemen-elemen dalam array. Standar deviasi dihitung dengan menghitung selisih setiap elemen terhadap nilai rata-rata, mengkuadratkan hasilnya, dan kemudian mengambil akar kuadrat dari rata-rata hasil tersebut. Pengguna juga dapat mengetahui frekuensi kemunculan suatu bilangan tertentu dalam array dengan memasukkan bilangan yang ingin dicari. Program kemudian menghitung berapa kali bilangan tersebut muncul dalam array.

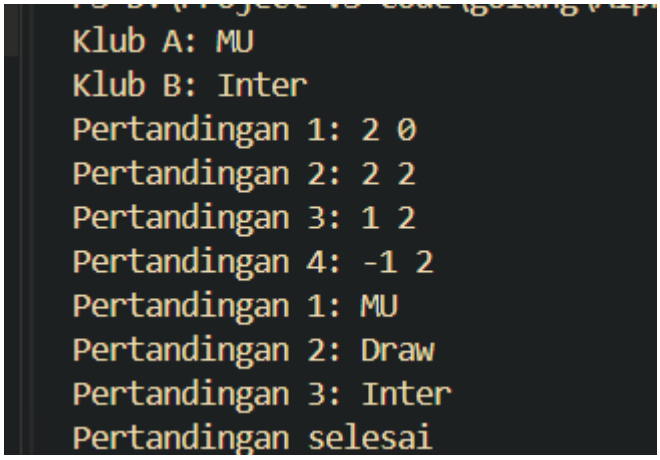
Dengan fitur-fitur ini, program memungkinkan pengguna untuk melakukan berbagai analisis statistik sederhana pada data yang disimpan dalam array, termasuk menemukan pola seperti kemunculan elemen genap atau ganjil, menghitung rata-rata dan standar deviasi, serta menampilkan elemen pada indeks tertentu.

3. SOURCE CODE

```go
package main

import "fmt"

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

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

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

    if skorA < 0 || skorB < 0 {
```

```
      break
    }

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

    pertandingan++
  }

  for i, winner := range pemenang {
    if winner != "Draw" {
      fmt.Printf("Pertandingan %d: %s\n", i+1, winner)
    } else {
      fmt.Printf("Pertandingan %d: Draw\n", i+1)
    }
  }
  fmt.Print("Pertandingan selesai")
}
```

OUTPUT



DESKRIPSI PROGRAM

Program ini adalah simulasi pencatatan hasil pertandingan antara dua klub sepak bola, yaitu Klub A dan Klub B. Program meminta pengguna untuk memasukkan nama kedua klub, kemudian memulai proses pencatatan hasil pertandingan. Setiap pertandingan

dinilai berdasarkan skor yang dimasukkan untuk masing-masing klub. Jika skor Klub A lebih tinggi, maka Klub A dinyatakan sebagai pemenang; jika skor Klub B lebih tinggi, maka Klub B dinyatakan pemenang. Jika skornya sama, pertandingan dianggap "Draw" atau seri.

Program ini menggunakan loop for untuk terus menerima skor setiap pertandingan hingga pengguna memasukkan skor negatif untuk salah satu klub. Skor negatif berfungsi sebagai tanda untuk mengakhiri input pertandingan. Selama loop berjalan, hasil setiap pertandingan disimpan dalam slice pemenang, yang menyimpan nama klub pemenang atau "Draw" untuk pertandingan yang berakhir seri.

Setelah pengguna berhenti memasukkan skor, program menampilkan hasil setiap pertandingan dengan loop lain. Loop ini menampilkan informasi mengenai pemenang atau status seri dari setiap pertandingan yang telah dicatat, diikuti dengan pesan "Pertandingan selesai" sebagai tanda akhir program.

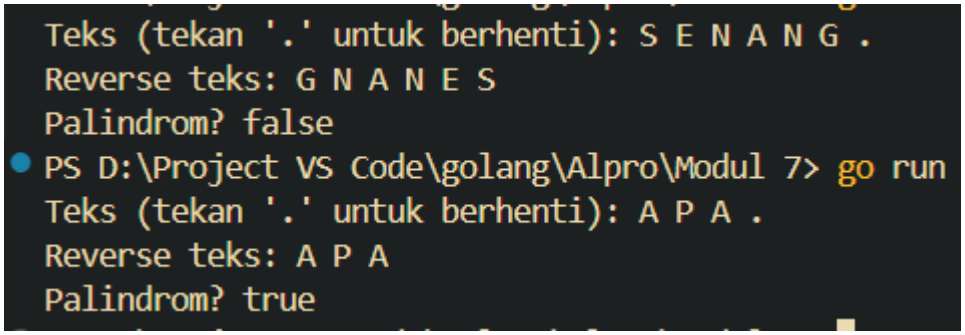
Dengan struktur ini, program memungkinkan pengguna untuk mencatat dan melihat hasil dari beberapa pertandingan berurutan antara dua klub secara sederhana.

4. SOURCE CODE

```go
package main

import "fmt"

const NMAX = 127

type tabel [NMAX]rune

func isiArray(t *tabel) int {
  var m int
  fmt.Print("Teks (tekan '.' untuk berhenti): ")
  for {
    var input string
    fmt.Scan(&input)
    if input == "." {
      break
    }
    t[m] = []rune(input)[0]
    m++
  }
  return m
}

func balikanArray(t *tabel, n int) {
  for i, j := 0, n-1; i < j; i, j = i+1, j-1 {
    t[i], t[j] = t[j], t[i]
  }
}

func cetakArray(t tabel, n int) {
  fmt.Print("Reverse teks: ")
  for i := 0; i < n; i++ {
    fmt.Print(string(t[i]), " ")
  }
  fmt.Println()
}

func palindrom(t tabel, n int) bool {
  for i := 0; i < n/2; i++ {
    if t[i] != t[n-i-1] {
      return false
    }
  }
  return true
}
```

```
func main() {
  var tab tabel
  m := isiArray(&tab)
  balikanArray(&tab, m)
  cetakArray(tab, m)

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

OUTPUT



DESKRIPSI PROGRAM

Program ini meminta input berupa karakter satu per satu dari pengguna hingga pengguna memasukkan titik (.) sebagai tanda berhenti. Setiap karakter dimasukkan ke dalam array tabel. Setelah itu, program membalikkan urutan karakter dalam array menggunakan fungsi balikanArray. Fungsi ini bekerja dengan menukar elemen pertama dengan yang terakhir, elemen kedua dengan yang kedua dari belakang, dan seterusnya. Kemudian, program mencetak array yang telah dibalik.

Setelah mencetak teks terbalik, program memeriksa apakah teks yang dimasukkan adalah palindrom atau bukan menggunakan fungsi palindrom. Fungsi ini membandingkan karakter-karakter dari awal dan akhir, jika ada pasangan karakter yang tidak sama, maka teks tersebut bukan palindrom.

Program kemudian menampilkan apakah teks yang dimasukkan membentuk palindrom atau tidak.