LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XI

NILAI EXTRIM

Disusun Oleh :

RAKHA YUDHISTIRA / 2311102010

IF-11-06

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Nilai ekstrem dalam pemrograman, termasuk dalam bahasa Go, adalah nilai minimum dan maksimum dari sekumpulan data yang diolah. Untuk menemukan nilai-nilai tersebut, diperlukan perbandingan elemen-elemen dalam suatu array atau slice. Proses ini melibatkan iterasi untuk memeriksa setiap elemen dan membandingkannya dengan nilai terkecil atau terbesar yang telah ditemukan sebelumnya.

Dalam Go, algoritma untuk menghitung nilai ekstrem biasanya melibatkan langkah-langkah berikut:

1. Inisialisasi nilai awal minimum dan maksimum dengan elemen pertama array/slice.
2. Iterasi melalui array untuk membandingkan setiap elemen:
    - Jika elemen lebih kecil dari nilai minimum saat ini, perbarui nilai minimum.
    - Jika elemen lebih besar dari nilai maksimum saat ini, perbarui nilai maksimum.
3. Hasil akhirnya adalah nilai minimum dan maksimum dari array tersebut.

II. GUIDED

1. **Soal Studi Case**

Sourcecode

```
package main

import "fmt"

// Mendeklarasikan tipe data array arrInt dengan panjang
2023
type arrInt [2023]int

// Fungsi untuk mencari indeks elemen terkecil dalam
array
func terkecil(tabInt arrInt, n int) int {
    var idx int = 0  // idx menyimpan indeks elemen
terkecil
    var j int = 1
    for j < n {
        if tabInt[idx] > tabInt[j] {
```

```
            idx = j  // Simpan indeks j jika elemen di
indeks j lebih kecil
        }
        j = j + 1
    }
    return idx
}

// Fungsi main untuk menguji fungsi terkecil
func main() {
    var n int
    var tab arrInt

    // Meminta input jumlah elemen array
    fmt.Print("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n)

    // Validasi input jumlah elemen
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah elemen harus antara 1 dan
2023.")
        return
    }

    // Memasukkan elemen-elemen array
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
        fmt.Print("Elemen ke-", i+1, ": ")
        fmt.Scan(&tab[i])
    }

    // Memanggil fungsi terkecil untuk menemukan indeks
elemen terkecil
    idxMin := terkecil(tab, n)

    // Menampilkan nilai dan indeks terkecil
    fmt.Println("Nilai terkecil dalam array adalah:",
tab[idxMin], "pada indeks:", idxMin)
}
```

Screenshoot Output

```
PS D:\ITTP\Semester 3\Alpro 2\2311102010_Rakha Yudhistira_Modul_11>
tira_Modul_11\guided1.go"
Masukkan jumlah elemen (maks 2023): 3
Masukkan elemen-elemen array:
Elemen ke-1: 2
Elemen ke-2: 3
Elemen ke-3: 4
Nilai terkecil dalam array adalah: 2 pada indeks: 0
```

Deskripsi Program

Program dibuat untuk mencari indeks elemen terkecil dalam sebuah array. Program mendeklarasikan tipe data array arrInt dengan panjang maksimal 2023 dan menggunakan fungsi terkecil untuk menemukan indeks elemen terkecil dalam array. Fungsi main meminta pengguna memasukkan jumlah elemen array (dengan validasi agar berada dalam rentang 1-2023), lalu menerima input nilai elemen-elemen array. Setelah itu, program memanggil fungsi terkecil untuk menghitung indeks elemen dengan nilai terkecil, dan menampilkan nilai serta indeks elemen tersebut ke layar.

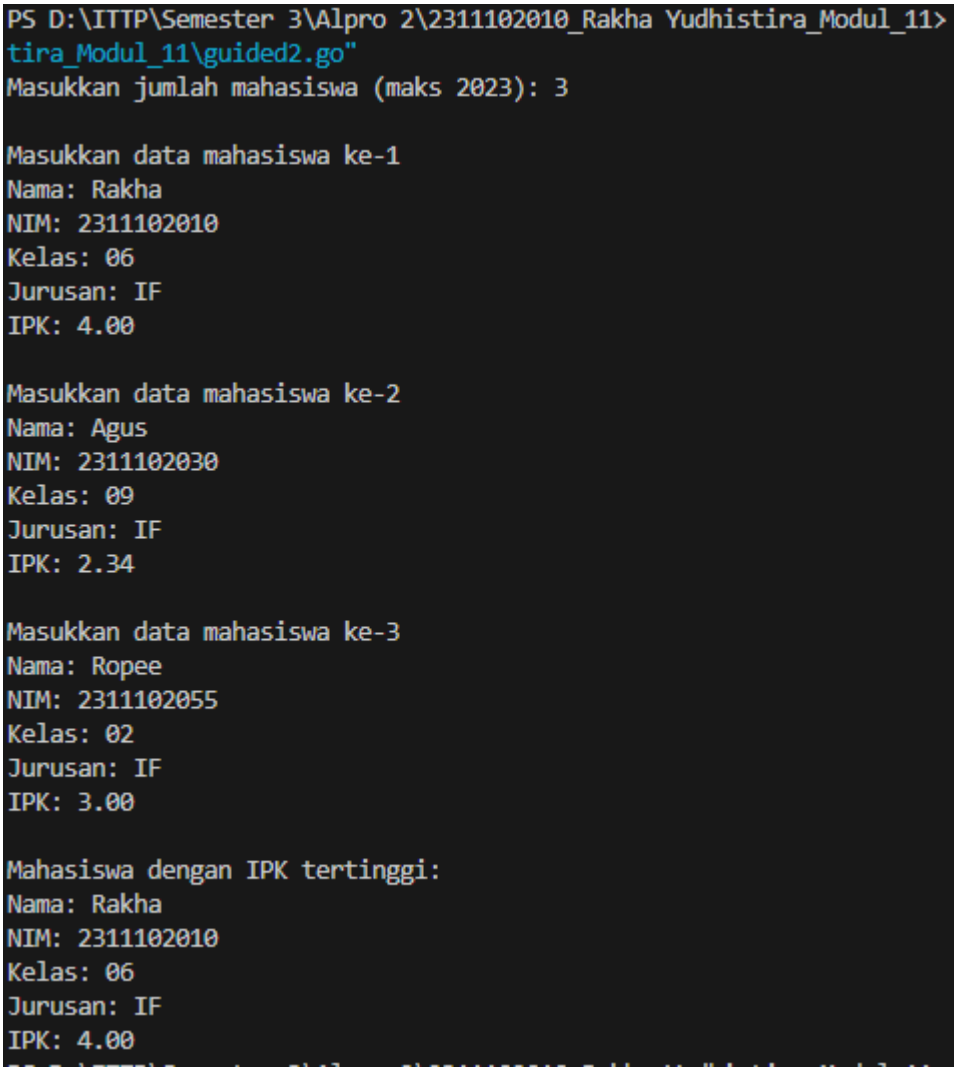
2. **Soal Studi Case**

Sourcecode

```
package main

import "fmt"

// Definisi struct mahasiswa dengan atribut nama, nim,
kelas, jurusan, dan ipk
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                       float64
}

// Definisi tipe data array mahasiswa dengan kapasitas
maksimal 2023
type arrMhs [2023]mahasiswa

// Fungsi untuk mencari indeks IPK tertinggi dalam array
mahasiswa
func indeksIPKTertinggi(T arrMhs, n int) int {
    var idx int = 0  // Inisialisasi indeks IPK
tertinggi pada indeks pertama
    for j := 1; j < n; j++ {
        if T[idx].ipk < T[j].ipk {
            idx = j // Update indeks jika ditemukan IPK
yang lebih tinggi
        }
    }
    return idx
}

// Fungsi main untuk mengisi data mahasiswa dan mencari
IPK tertinggi beserta indeksnya
func main() {
    var n int
    var dataMhs arrMhs

    // Meminta input jumlah mahasiswa
```

```
    fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")
    fmt.Scan(&n)

    // Validasi jumlah mahasiswa yang dimasukkan
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah mahasiswa harus antara 1 dan
2023.")
        return
    }

    // Mengisi data mahasiswa
    for i := 0; i < n; i++ {
        fmt.Printf("\nMasukkan data mahasiswa ke-%d\n",
i+1)
        fmt.Print("Nama: ")
        fmt.Scan(&dataMhs[i].nama)
        fmt.Print("NIM: ")
        fmt.Scan(&dataMhs[i].nim)
        fmt.Print("Kelas: ")
        fmt.Scan(&dataMhs[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scan(&dataMhs[i].jurusan)
        fmt.Print("IPK: ")
        fmt.Scan(&dataMhs[i].ipk)
    }

    // Mendapatkan indeks IPK tertinggi
    idxTertinggi := indeksIPKTertinggi(dataMhs, n)

    // Menampilkan data mahasiswa dengan IPK tertinggi
    fmt.Printf("\nMahasiswa dengan IPK tertinggi:\n")
    fmt.Printf("Nama: %s\n", dataMhs[idxTertinggi].nama)
    fmt.Printf("NIM: %s\n", dataMhs[idxTertinggi].nim)
    fmt.Printf("Kelas: %s\n",
dataMhs[idxTertinggi].kelas)
    fmt.Printf("Jurusan: %s\n",
dataMhs[idxTertinggi].jurusan)
    fmt.Printf("IPK: %.2f\n", dataMhs[idxTertinggi].ipk)
}
```

Screenshoot Output



Deskripsi Program

Program dibuat untuk mencari mahasiswa dengan IPK tertinggi dari data yang dimasukkan pengguna. Program menggunakan struct mahasiswa untuk merepresentasikan data mahasiswa yang meliputi atribut nama, NIM, kelas, jurusan, dan IPK, serta tipe data array arrMhs dengan kapasitas maksimal 2023 untuk menyimpan data mahasiswa. Fungsi indeksIPKTertinggi digunakan untuk menemukan indeks mahasiswa dengan IPK tertinggi dalam array. Fungsi main meminta pengguna untuk menginput jumlah mahasiswa (dengan validasi), kemudian mengisi data

setiap mahasiswa. Setelah itu, program mencari dan menampilkan informasi mahasiswa dengan IPK tertinggi, termasuk nama, NIM, kelas, jurusan, dan IPK.

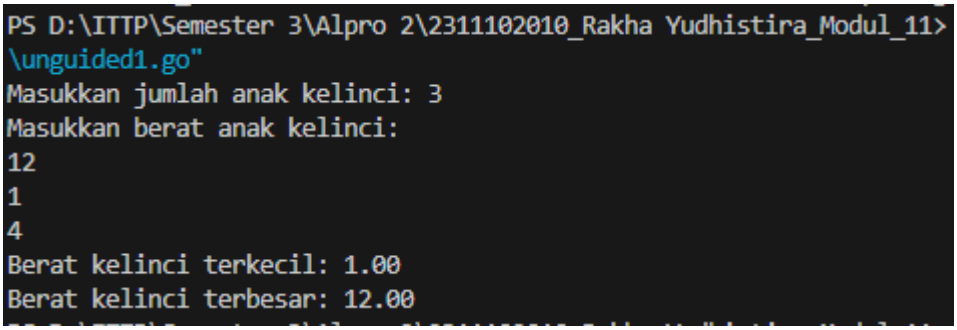
III. UNGUIDED

1. **Soal Studi Case**

Sourcecode

```
// Rakha Yudhistira_2311102010_IF-11-06

package main

import "fmt"

func main4() {
    var N int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&N)

    if N <= 0 || N > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1
dan 1000")
        return
    }

    weights := make([]float64, N)
    fmt.Println("Masukkan berat anak kelinci:")
    for i := 0; i < N; i++ {
        fmt.Scan(&weights[i])
    }

    minWeight, maxWeight := weights[0], weights[0]

    for _, weight := range weights {
        if weight < minWeight {
            minWeight = weight
        }
        if weight > maxWeight {
            maxWeight = weight
        }
    }

    fmt.Printf("Berat kelinci terkecil: %.2f\n",
minWeight)
    fmt.Printf("Berat kelinci terbesar: %.2f\n",
maxWeight)
}
```

Screenshoot Output



Deskripsi Program

Program dibuat untuk menerima input jumlah dan berat anak kelinci, lalu menentukan berat terkecil dan terbesar. Pengguna diminta memasukkan jumlah anak kelinci (N) dengan validasi agar berada dalam rentang 1 hingga 1000. Berat masing-masing kelinci dimasukkan ke dalam slice weights. Program kemudian menginisialisasi berat terkecil (minWeight) dan terbesar (maxWeight) dengan nilai berat pertama, lalu menggunakan perulangan untuk membandingkan setiap berat, memperbarui nilai terkecil atau terbesar sesuai kondisi. Akhirnya, program mencetak berat kelinci terkecil dan terbesar dalam format desimal dua angka.

2. **Soal Studi Case**

Sourcecode

```
// Rakha Yudhistira_2311102010_IF-11-06

package main

import "fmt"

const maxCapacity = 1000

func calculateWeight(x int, y int, weights
[maxCapacity]float64) ([]float64, float64) {
    numContainers := (x + y - 1) / y
    containerWeights := make([]float64, numContainers)

    for i := 0; i < x; i++ {
        containerIndex := i / y
        containerWeights[containerIndex] += weights[i]
    }

    totalWeight_2311102010 := 0.0
    for _, weight := range containerWeights {
        totalWeight_2311102010 += weight
```

```
    }
    averageWeight := totalWeight_2311102010 /
float64(numContainers)

    return containerWeights, averageWeight
}

func main() {
    var x, y int
    var weights [maxCapacity]float64

    fmt.Print("Masukkan jumlah ikan (x) dan kapasitas
wadah (y): ")
    fmt.Scan(&x, &y)

    if x > maxCapacity {
        fmt.Printf("Jumlah ikan melebihi kapasitas
maksimum %d\n", maxCapacity)
        return
    }

    fmt.Printf("Masukkan berat ikan sebanyak %d:\n", x)
    for i := 0; i < x; i++ {
        fmt.Printf("Berat ikan ke-%d: ", i+1)
        fmt.Scan(&weights[i])
    }

    containerWeights, averageWeight :=
calculateWeight(x, y, weights)

    fmt.Println("\nHASIL PERHITUNGAN IKAN")
    fmt.Println("Total berat ikan di setiap wadah adalah
sebagai berikut:")
    for i, weight := range containerWeights {
        fmt.Printf("Wadah %d: %.2f kg\n", i+1, weight)
    }

    fmt.Printf("\nBerat rata-rata ikan di setiap wadah
adalah %.2f kg\n", averageWeight)
}
```

Screenshoot Output

```
PS D:\ITTP\Semester 3\Alpro 2\2311102010_Rakha Yudhistira_Modul_11>
\unguided2.go"
Masukkan jumlah ikan (x) dan kapasitas wadah (y): 3 2
Masukkan berat ikan sebanyak 3:
Berat ikan ke-1: 23
Berat ikan ke-2: 12
Berat ikan ke-3: 31

HASIL PERHITUNGAN IKAN
Total berat ikan di setiap wadah adalah sebagai berikut:
Wadah 1: 35.00 kg
Wadah 2: 31.00 kg

Berat rata-rata ikan di setiap wadah adalah 33.00 kg
```

Deskripsi Program

Program dibuat untuk menghitung total berat ikan di setiap wadah dan rata-rata berat ikan per wadah berdasarkan jumlah ikan (x) dan kapasitas wadah (y). Pengguna diminta memasukkan jumlah ikan, kapasitas wadah, dan berat masing-masing ikan. Fungsi calculateWeight menghitung jumlah wadah yang diperlukan dan mendistribusikan berat ikan ke wadah-wadah tersebut. Total berat ikan di setiap wadah disimpan dalam slice containerWeights, lalu dihitung rata-ratanya. Hasil akhirnya mencetak total berat ikan per wadah dan berat rata-rata di setiap wadah dalam format dua desimal.

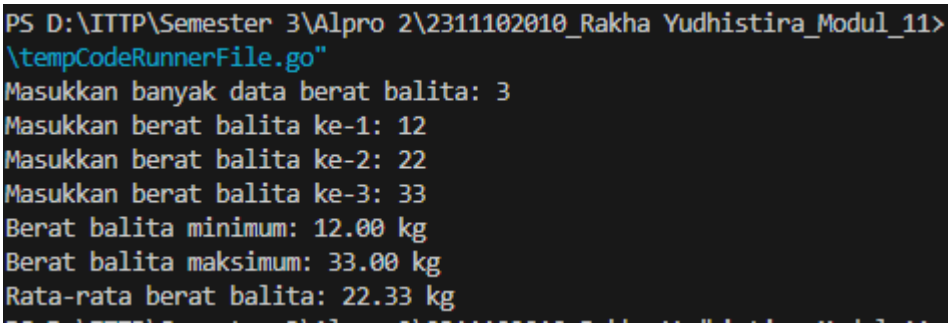
3. **Soal Studi Case**

Sourcecode

```
// Rakha Yudhistira_2311102010_IF-11-06

package main

import (
    "fmt"
)

type arrBalita [100]float64

func hitungMinMax_2311102010(arr arrBalita, n int, min,
max *float64) {
    *min = arr[0]
    *max = arr[0]
    for i := 1; i < n; i++ {
        if arr[i] < *min {
```

```
            *min = arr[i]
        }
        if arr[i] > *max {
            *max = arr[i]
        }
    }
}
func rataRata(arr arrBalita, n int) float64 {
    var total float64 = 0
    for i := 0; i < n; i++ {
        total += arr[i]
    }
    return total / float64(n)
}
func main() {
    var n int
    var berat arrBalita
    var min, max float64
    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&n)
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
        fmt.Scan(&berat[i])
    }
    hitungMinMax_2311102010(berat, n, &min, &max)
    rata := rataRata(berat, n)
    // Output hasil
    fmt.Printf("Berat balita minimum: %.2f kg\n", min)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", max)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n",
rata)
}
```

Screenshoot Output



Deskripsi Program

Program ini dibuat untuk menghitung berat minimum, maksimum, dan rata-rata dari data berat balita. Program menggunakan tipe data arrBalita untuk menyimpan berat balita dengan kapasitas maksimal 100. Fungsi hitungMinMax_2311102010 menerima array berat balita, menghitung berat minimum dan maksimum, lalu menyimpannya melalui pointer. Fungsi rataRata menghitung rata-rata dengan menjumlahkan semua berat dan membaginya dengan jumlah data. Pada fungsi main, program meminta input jumlah data berat balita (n) dan berat masing-masing balita, lalu memanggil kedua fungsi tersebut untuk melakukan perhitungan. Hasil akhirnya menampilkan berat minimum, maksimum, dan rata-rata balita dalam format desimal dua angka.