

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 7
MATERI**



Oleh:

DAFFA TSAQIFNA FAUZTSANY

103112400032

S1 IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

I. DASAR TEORI

Pencarian Nilai Ekstrim

1. Pengertian

Pencarian nilai ekstrim adalah proses mencari nilai terbesar (maksimum) atau terkecil (minimum) dari sekumpulan data, misalnya pada array.

2. Langkah Umum Algoritma:

- Asumsikan data pertama sebagai nilai ekstrim.
- Bandingkan dengan elemen berikutnya.
- Jika ada yang lebih ekstrim, update nilai tersebut.
- Setelah seluruh data dicek, hasil akhir adalah nilai ekstrimnya.

Contoh Pencarian Minimum:

```
func terkecil(tabInt [100]int, n int) int {  
    min := tabInt[0]  
    for i := 1; i < n; i++ {  
        if tabInt[i] < min {  
            min = tabInt[i]  
        }  
    }  
    return min  
}
```

Pencarian pada Data Terstruktur (Struct)

Bisa juga dilakukan pada array of struct, seperti mencari mahasiswa dengan IPK tertinggi.

```
type mahasiswa struct {  
    nama string  
    ipk float64  
}  
  
func cariMaxIPK(data []mahasiswa) int {  
    idx := 0  
    for i := 1; i < len(data); i++ {  
        if data[i].ipk > data[idx].ipk {  
            idx = i  
        }  
    }  
    return idx  
}
```

3. Catatan Penting:

- Bisa mencari nilai atau indeks dari nilai tersebut.
- Cocok digunakan untuk data bertipe dasar dan kompleks.
- Gunakan loop sederhana dan logika perbandingan.

II. GUIDED

Source Code + Screenshot hasil program beserta penjelasan

1. GUIDED 1

Source Code:

```
package main

import (
    "fmt"
)

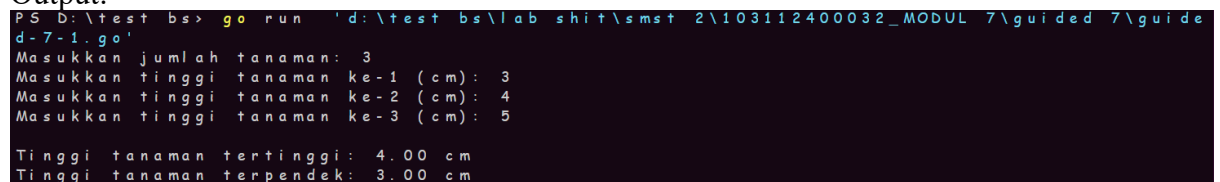
func main() {
    var n int
    fmt.Print("Masukkan jumlah tanaman: ")
    fmt.Scan(&n)

    var tinggiTanaman [500]float64
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan tinggi tanaman ke-%d (cm): ", i+1)
        fmt.Scan(&tinggiTanaman[i])
    }

    min, max := tinggiTanaman[0], tinggiTanaman[0]
    for i := 1; i < n; i++ {
        if tinggiTanaman[i] < min {
            min = tinggiTanaman[i]
        }
        if tinggiTanaman[i] > max {
            max = tinggiTanaman[i]
        }
    }

    fmt.Printf("\nTinggi tanaman tertinggi: %.2f cm\n", max)
    fmt.Printf("Tinggi tanaman terpendek: %.2f cm\n", min)
}
```

Output:



```
PS D:\test bs> go run 'd:\test bs\lab shit\smsr 2\103112400032_MODUL 7\guided 7\guide d-7-1.go'
Masukkan jumlah tanaman: 3
Masukkan tinggi tanaman ke-1 (cm): 3
Masukkan tinggi tanaman ke-2 (cm): 4
Masukkan tinggi tanaman ke-3 (cm): 5

Tinggi tanaman tertinggi: 4.00 cm
Tinggi tanaman terpendek: 3.00 cm
```

Deskripsi Program:

Program ini digunakan untuk mencatat tinggi beberapa tanaman dan menentukan tinggi maksimum (tertinggi) serta tinggi minimum (terpendek) di antara data yang dimasukkan. Pengguna diminta memasukkan jumlah tanaman, lalu tinggi masing-masing tanaman. Program menyimpan data dalam array, kemudian membandingkan setiap nilai untuk

mendapatkan hasil tertinggi dan terendah. Hasil akhir ditampilkan dalam satuan sentimeter dengan dua angka di belakang koma.

2. GUIDED 2

Source Code:

```
package main

import (
    "fmt"
)

func main() {
    var x, y int
    fmt.Print("Masukkan jumlah buku dan jumlah buku per rak: ")
    fmt.Scan(&x, &y)

    var hargaBuku [500]float64
    fmt.Println("\nMasukkan harga setiap buku (dalam ribuan Rp): ")
    for i := 0; i < x; i++ {
        fmt.Scan(&hargaBuku[i])
    }

    var hargaRataRata []float64
    for i := 0; i < x; i += y {
        total := 0.0
        for j := i; j < i+y && j < x; j++ {
            total += hargaBuku[j]
        }
        hargaRataRata = append(hargaRataRata, total/float64(y))
    }

    min, max := hargaBuku[0], hargaBuku[0]
    for _, harga := range hargaBuku[:x] {
        if harga < min {
            min = harga
        }
        if harga > max {
            max = harga
        }
    }

    fmt.Printf("\nRata-rata harga per rak: ")
    for _, avg := range hargaRataRata {
        fmt.Printf("%.2f ", avg)
    }
    fmt.Printf("\nHarga termahal: %.2f Rp\n", max)
    fmt.Printf("Harga termurah: %.2f Rp\n", min)
}
```

Output:

```
PS D:\test bs> go run 'd:\test bs\lab shit\sms 2\103112400032_MODUL 7\guided 7\guide
d-7-2.go'
Masukkan jumlah buku dan jumlah buku per rak: 3 4

Masukkan harga setiap buku (dalam ribuan Rp):
3000
30000
300000

Rata-rata harga per rak: 83250.00
Harga termahal: 300000.00 Rp
Harga termurah: 3000.00 Rp
```

Deskripsi Program:

Program ini digunakan untuk menghitung rata-rata harga buku per rak, serta menentukan harga buku termahal dan termurah. Pengguna memasukkan jumlah buku dan jumlah buku per rak, lalu memasukkan harga tiap buku (dalam ribuan rupiah). Program mengelompokkan buku ke dalam rak, menghitung rata-rata harga pada setiap rak, dan mencari nilai maksimum dan minimum dari seluruh harga buku yang dimasukkan. Hasil akhir ditampilkan dalam format desimal dua angka di belakang koma.

3. GUIDED 3

Source Code:

```
package main

import (
    "fmt"
)

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

    var nilaiSiswa [200]float64
    var totalNilai float64 = 0

    fmt.Println("\nMasukkan nilai ujian masing-masing siswa: ")
    for i := 0; i < n; i++ {
        fmt.Scan(&nilaiSiswa[i])
        totalNilai += nilaiSiswa[i]
    }

    rataRata := totalNilai / float64(n)

    min, max := nilaiSiswa[0], nilaiSiswa[0]
    var diAtasRataRata int = 0
    for _, nilai := range nilaiSiswa[:n] {
        if nilai < min {
            min = nilai
        }
    }
```

```

        if nilai > max {
            max = nilai
        }
        if nilai > rataRata {
            diAtasRataRata++
        }
    }
    fmt.Printf("\nNilai terendah: %.0f\n", min)
    fmt.Printf("Nilai tertinggi: %.0f\n", max)
    fmt.Printf("Rata-rata kelas: %.2f\n", rataRata)
    fmt.Printf("Jumlah siswa di atas rata-rata: %d\n", diAtasRataRata)
}

```

Output:

```

PS D:\test bs> go run 'd:\test bs\lab shit\smsr 2\103112400032_MODUL 7\guided 7\guide
d-7-3.go'
Masukkan jumlah siswa: 4

Masukkan nilai ujian masing-masing siswa:
50
80
70
90

Nilai terendah: 50
Nilai tertinggi: 90
Rata-rata kelas: 72.50
Jumlah siswa di atas rata-rata: 2

```

Deskripsi Program:

Program ini digunakan untuk memproses dan menganalisis nilai ujian siswa. Setelah pengguna memasukkan jumlah siswa dan nilai masing-masing, program menghitung rata-rata nilai kelas, lalu menentukan nilai tertinggi, nilai terendah, dan jumlah siswa yang nilainya di atas rata-rata. Hasil akhir ditampilkan dengan format rapi dan presisi yang sesuai.

III. UNGUIDED

Source Code + Screenshot hasil program beserta penjelasan

1. UNGUIDED 1

Source Code:

```
package main

import "fmt"

type KelinciData struct {
    weights []float64
    count   int
}

const maxSize = 1000

func inputBeratKelinci() KelinciData {
    var n int

    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&n)

    if n <= 0 || n > maxSize {
        fmt.Println("Jumlah anak kelinci tidak valid (1 - 1000).")
        return KelinciData{}
    }

    berat := make([]float64, n)
    fmt.Printf("Masukkan %d berat anak kelinci:\n", n)
    for i := 0; i < n; i++ {
        fmt.Scan(&berat[i])
    }

    return KelinciData{
        weights: berat,
        count:   n,
    }
}

func cariMinMax(data KelinciData) (float64, float64) {
    if data.count == 0 {
        return 0, 0
    }

    min := data.weights[0]
    max := data.weights[0]

    for i := 1; i < data.count; i++ {
        if data.weights[i] < min {
```

```

        min = data.weights[i]
    }
    if data.weights[i] > max {
        max = data.weights[i]
    }
}

return min, max
}

func main() {
    data := inputBeratKelinci()

    if data.count == 0 {
        return
    }

    min, max := cariMinMax(data)

    fmt.Printf("Berat terkecil: %.2f\n", min)
    fmt.Printf("Berat terbesar: %.2f\n", max)
}

```

Output:

```

PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MOBUL 7\unguided 7\unguided-7-1.go'
Masukkan jumlah anak kelinci: 3
Masukkan 3 berat anak kelinci:
4
5
6
Berat terkecil: 4.00
Berat terbesar: 6.00

```

Deskripsi Program:

Program ini digunakan untuk mencatat dan menganalisis berat anak kelinci, dengan batas maksimum data hingga 1000 ekor. Setelah pengguna memasukkan jumlah kelinci dan berat masing-masing, program menyimpan data dalam slice dan menggunakan fungsi cariMinMax untuk menentukan berat terkecil dan terbesar. Hasilnya ditampilkan dengan format dua angka desimal. Program juga memvalidasi jumlah input agar berada dalam rentang yang diperbolehkan.

2. UNGUIDED 2

Source Code:

```

package main

import "fmt"

type IkanData struct {
    berat    []float64
    jumlahIkan int
    isiPerWadah int
}

```



```

}

const maxSize = 1000

func inputIkan() IkanData {
    var x, y int

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

    if x <= 0 || x > maxSize || y <= 0 {
        fmt.Println("Input tidak valid.")
        return IkanData{}
    }

    arr := make([]float64, x)
    fmt.Printf("Masukkan %d berat ikan:\n", x)
    for i := 0; i < x; i++ {
        fmt.Scan(&arr[i])
    }

    return IkanData{
        berat:    arr,
        jumlahIkan: x,
        isiPerWadah: y,
    }
}

func hitungTotalPerWadah(data IkanData) []float64 {
    var hasil []float64
    jumlah := data.jumlahIkan
    kaps := data.isiPerWadah

    for i := 0; i < jumlah; i += kaps {
        total := 0.0
        for j := i; j < i+kaps && j < jumlah; j++ {
            total += data.berat[j]
        }
        hasil = append(hasil, total)
    }

    return hasil
}

func hitungRataRata(wadah []float64) float64 {
    if len(wadah) == 0 {
        return 0
    }
    total := 0.0
    for _, val := range wadah {

```

```

        total += val
    }
    return total / float64(len(wadah))
}

func main() {
    data := inputIkan()
    if data.jumlahIkan == 0 {
        return
    }

    wadah := hitungTotalPerWadah(data)
    for _, berat := range wadah {
        fmt.Printf("%.2f ", berat)
    }
    fmt.Println()

    rata := hitungRataRata(wadah)
    fmt.Printf("%.2f\n", rata)
}

```

Output:

```

PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 7\unguided 7\unguided-7-2.go'
Masukkan jumlah ikan dan kapasitas wadah (x y): 3 4
Masukkan 3 berat ikan:
4
5
6
15.00
15.00
PS D:\test bs> go run 'd:\test bs\lab shit\smst 2\103112400032_MODUL 7\unguided 7\unguided-7-2.go'
Masukkan jumlah ikan dan kapasitas wadah (x y): 3 4
Masukkan 3 berat ikan:
3
3
3
9.00
9.00

```

Deskripsi Program:

Program ini digunakan untuk mengelompokkan berat ikan ke dalam beberapa wadah sesuai kapasitas yang ditentukan, kemudian menghitung total berat per wadah serta rata-rata total berat antar semua wadah. Pengguna memasukkan jumlah ikan dan kapasitas tiap wadah, lalu memasukkan berat masing-masing ikan. Program menghitung total berat ikan dalam tiap wadah secara berurutan, menyimpan hasilnya dalam slice, dan mencetak seluruh total serta nilai rata-ratanya dengan dua angka di belakang koma.

3. UNGUIDED 3

Source Code:

```

package main

import "fmt"

func biaya(rp int) int {

```

```

var x, y int
x = rp / 1000
x *= 10000
if rp%1000 != 0 {
    y = rp - (x / 10)
    if y >= 500 {
        y *= 5
        return x + y
    } else {
        y *= 15
        return x + y
    }
} else {
    return x
}
}
func main() {
    var x, y int
    fmt.Print("Berat parsel (gram): ")
    fmt.Scan(&x)
    y = x % 1000
    if y >= 500 {
        y *= 5
    } else {
        y *= 15
    }
    fmt.Printf("Detail berat: %d kg + %d gr\n", x/1000, x%1000)
    fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", (x/1000)*10000, y)
    fmt.Printf("Total biaya: Rp. %d\n", biaya(x))
}

```

Output:

```

PS D:\test bs> go run 'd:\test bs\lab shit\smsr 2\103112400032_MODUL 7\unguided 7\unguided-7-3.go'
Masukan banyak data berat balita: 3
Masukan berat balita ke-1: 6
Masukan berat balita ke-2: 7
Masukan berat balita ke-3: 8

Berat balita minimum: 6.00 kg
Berat balita maksimum: 8.00 kg
Rerata berat balita: 7.00 kg
Jumlah balita dengan berat di atas rata-rata: 1

```

Deskripsi Program:

Program ini digunakan untuk menganalisis data berat balita. Pengguna memasukkan jumlah data dan berat masing-masing balita, kemudian program menghitung dan menampilkan:

1. Berat minimum dan maksimum menggunakan fungsi dengan parameter pointer.
2. Rata-rata berat seluruh balita.
3. Jumlah balita yang beratnya di atas rata-rata.

Semua data disimpan dalam array statis dan diproses menggunakan fungsi terpisah untuk modularitas. Hasil akhir ditampilkan dengan format dua angka desimal.