

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XI  
NILAI EKSTRIM**



**Disusun Oleh :**

**Marsep Trianto Pakondo / 2311102251**

**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 ekstrim dalam himpunan data merujuk pada nilai minimum dan maksimum yang terdapat dalam himpunan tersebut. Identifikasi nilai-nilai ini sangat penting dalam berbagai aplikasi komputasi seperti pengolahan data statistik, pengelolaan data besar, dan pemrosesan sinyal.

### **1. Definisi Nilai Ekstrim**

- **Nilai Minimum:** Elemen terkecil dalam suatu himpunan data.
- **Nilai Maksimum:** Elemen terbesar dalam suatu himpunan data.

### **2. Algoritma Pencarian**

Algoritma pencarian nilai ekstrem biasanya dilakukan melalui iterasi sederhana:

1. Menginisialisasi nilai minimum dan maksimum dengan elemen pertama dalam himpunan.
2. Membandingkan setiap elemen lainnya terhadap nilai minimum dan maksimum yang tersimpan.
3. Memperbarui nilai jika ditemukan elemen yang lebih kecil (untuk minimum) atau lebih besar (untuk maksimum).

## II. GUIDED

### Soal Studi Case

XXXXXXXXXXXXXXXXXX

### 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
```

```

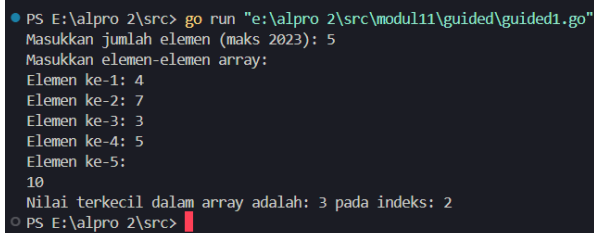
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
        fmt.Print("Elemen ke-", i+1, ": ")
        fmt.Scan(&tab[i])
    }

    // 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 E:\alpro 2\src> go run "e:\alpro 2\src\modul11\guided\guided1.go"
Masukkan jumlah elemen (maks 2023): 5
Masukkan elemen-elemen array:
Elemen ke-1: 4
Elemen ke-2: 7
Elemen ke-3: 3
Elemen ke-4: 5
Elemen ke-5:
10
Nilai terkecil dalam array adalah: 3 pada indeks: 2
PS E:\alpro 2\src>

```

## Deskripsi Program

Program diatas digunakan untuk untuk menemukan elemen terkecil dalam sebuah array bilangan bulat. Pengguna diminta memasukkan jumlah elemen yang akan diproses, kemudian memasukkan nilai setiap elemen secara berurutan. Setelah data dimasukkan, program mencari elemen dengan nilai terkecil dan menampilkan nilainya beserta indeksnya dalam array. Program ini menggunakan tipe data `arrInt`, yaitu array dengan kapasitas maksimal 2023, untuk menyimpan bilangan bulat. Fungsi `terkecil` bertugas membandingkan setiap elemen array guna menemukan elemen terkecil dengan memperbarui indeksnya saat ditemukan nilai yang lebih kecil. Fungsi utama (`main`) menangani input jumlah elemen, mengisi nilai-

*nilai elemen dalam array, memanggil fungsi terkecil, dan mencetak hasil berupa nilai terkecil serta indeksinya.*

### 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 IPK tertinggi dalam array mahasiswa
func ipk(T arrMhs, n int) float64 {
    var tertinggi float64 = T[0].ipk
    var j int = 1
    for j < n {
        if tertinggi < T[j].ipk {
            tertinggi = T[j].ipk
        }
        j = j + 1
    }
    return tertinggi
}

// Fungsi main untuk mengisi data mahasiswa dan mencari IPK
// tertinggi
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)
        }

        // Mencari dan menampilkan IPK tertinggi
        tertinggi := ipk(dataMhs, n)
        fmt.Printf("\nIPK tertinggi dari %d mahasiswa adalah:
%.2f\n", n, tertinggi)
    }

```

## Screenshoot Output

```
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul11\guided\guided2.go"
Masukkan jumlah mahasiswa (maks 2023): 3

Masukkan data mahasiswa ke-1
Nama: Asep
NIM: 2311102251
Kelas: IF-11-06
Jurusan: Informatika
IPK: 3.8

Masukkan data mahasiswa ke-2
Nama: Roy
NIM: 2311102278
Kelas: IF-11-01
Jurusan: Informatika
IPK: 3.67

Masukkan data mahasiswa ke-3
Nama: Cecep
NIM: 2311102244
Kelas: IF-11-07
Jurusan: Informatika
IPK: 3.78

IPK tertinggi dari 3 mahasiswa adalah: 3.80
PS E:\alpro 2\src>
```

## Deskripsi Program

*Program diatas digunakan untuk mengetahui nilai ipk tertinggi dari mahasiswa. Pengguna memasukkan informasi seperti nama, NIM, kelas, jurusan, dan IPK untuk sejumlah mahasiswa. Setelah data dimasukkan, program akan mencari dan menampilkan IPK tertinggi dari semua mahasiswa. Program ini menggunakan struktur data mahasiswa untuk menyimpan informasi setiap mahasiswa dan array arrMhs dengan kapasitas maksimal 2023 untuk menyimpan data seluruh mahasiswa. Fungsi ipk bertugas mencari dan mengembalikan nilai IPK tertinggi dari array data mahasiswa berdasarkan input jumlah mahasiswa. Fungsi utama (main) meminta pengguna memasukkan jumlah mahasiswa, kemudian menginput data setiap mahasiswa, memanggil fungsi ipk untuk menentukan IPK tertinggi, dan menampilkan hasilnya.*

### III. UNGUIDED

#### Soal Studi Case

XXXXXXXXXXXXXXXXXXXX

#### Sourcecode

```
package main

import "fmt"

type berat [1000]float64

func beratTerkecil(b berat, n int) float64 {
    var min float64 = b[0]
    for i := 1; i < n; i++ {
        if min > b[i] {
            min = b[i]
        }
    }

    return min
}

func beratTerbesar(b berat, n int) float64 {
    var maks float64 = b[0]
    for i := 1; i < n; i++ {
        if maks < b[i] {
            maks = b[i]
        }
    }

    return maks
}

func main() {
    var b berat
    var n int

    fmt.Print("Masukkan jumlah anak kelinci (maks 1000): ")
    fmt.Scan(&n)

    if n < 1 || n > 1000 {
```



```

        fmt.Println("Jumlah mahasiswa harus antara 1
dan 2023.")
    }
    return
}

fmt.Println("Masukkan berat dari anak kelinci :")
for i := 0; i < n; i++ {
    fmt.Print("Anak Kelinci ke-", i+1, ": ")
    fmt.Scan(&b[i])
}

terkecil := beratTerkecil(b, n)
terbesar := beratTerbesar(b, n)

fmt.Printf("Nilai berat kelinci terkecil : %.2f Kg\n", terkecil)
fmt.Printf("Nilai berat kelinci terbesar : %.2f Kg\n", terbesar)
}

```

## Screenshoot Output

```

PS E:\alpro 2\src> go run "E:\alpro 2\src\modul11\unguided\unguided1.go"
Masukkan jumlah anak kelinci (maks 1000): 4
Masukkan berat dari anak kelinci :
Anak Kelinci ke-1: 1.1
Anak Kelinci ke-2: 2
Anak Kelinci ke-3: 0.9
Anak Kelinci ke-4: 1.67
Nilai berat kelinci terkecil : 0.90 Kg
Nilai berat kelinci terbesar : 2.00 Kg
PS E:\alpro 2\src>

```

## Deskripsi Program

*Program diatas digunakan untuk mencari berat terkecil dan terbesar dari sekumpulan data berat anak kelinci. Pengguna diminta memasukkan jumlah anak kelinci, lalu memasukkan data berat masing-masing satu per satu. Program ini menggunakan tipe data berat, berupa array dengan kapasitas maksimal 1000, untuk menyimpan data berat dalam bentuk bilangan real. Fungsi beratTerkecil digunakan untuk mencari nilai berat terkecil, sementara fungsi beratTerbesar digunakan untuk mencari nilai berat terbesar. Fungsi utama (main) menangani input jumlah data dan berat tiap anak kelinci, memanggil kedua fungsi tersebut untuk menentukan nilai minimum dan maksimum, lalu menampilkan hasilnya kepada pengguna.*

## Sourcecode

```
package main

import "fmt"

type beratIkan [1000]float64

func totalBeratIkan(ikan beratIkan, x, y int) {
    var total float64
    wadah := 1
    fmt.Println("\nTotal berat ikan disetiap wadah : ")

    for i := 1; i <= x; i++ {
        if i % y == 0 {
            total += ikan[i-1]
            fmt.Printf("Wadah ke-%v : %.2f Kg\n",
wadah, total)
            wadah++
            total = 0.0
        } else {
            total += ikan[i-1]
            if i == x && i % y != 0 {
                fmt.Printf("Sisanya di wadah ke-
%v : %.2f Kg\n", wadah, total)
            }
        }
    }
}

func beratRataIkan(ikan beratIkan, x, y int) {
    var total, rata, bagi float64
    wadah := 1
    fmt.Println("\nRata-rata berat ikan disetiap wadah : ")

    for i := 1; i <= x; i++ {
        if i % y == 0 {
            total += ikan[i-1]
            bagi++
            rata = total / bagi
            fmt.Printf("Wadah ke-%v : %.2f Kg\n",
wadah, rata)
            wadah++
            total = 0.0
        }
    }
}
```

```

        rata = 0.0
        bagi = 0.0
    } else {
        total += ikan[i-1]
        bagi++
        if i == x && i % y != 0 {
            rata = total / bagi
            fmt.Printf("Wadah ke-%v : %.2f
Kg\n", wadah, rata)
        }
    }
}

func main() {
    var x, y int
    var ikan beratIkan

    fmt.Print("Masukkan banyaknya ikan dijual (maks 1000):
")
    fmt.Scan(&x)
    fmt.Printf("Masukkan banyaknya ikan yang dimasukkan
ke dalam wadah (maks sebanyak %v): ", x)
    fmt.Scan(&y)

    if x < 1 || x > 1000 {
        fmt.Println("Jumlah ikan yang dijual harus antara
1 dan 1000.")
        return
    }
    if y < 1 || y > x {
        fmt.Printf("Jumlah ikan yang dimasukkan ke
dalam wadah harus antara 1 dan %v.\n", x)
        return
    }

    fmt.Println("\nMasukkan berat ikan :")
    for i := 0; i < x; i++ {
        fmt.Print("Berat ikan ke-", i+1, ": ")
        fmt.Scan(&ikan[i])
    }

    totalBeratIkan(ikan, x, y)

```

```
beratRataIkan(ikan, x, y)
}
```

## Screenshoot Output

```
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul11\unguided\unguided2.go"
• Masukkan banyaknya ikan dijual (maks 1000): 6
Masukkan banyaknya ikan yang dimasukkan ke dalam wadah (maks sebanyak 6): 4

Masukkan berat ikan :
Berat ikan ke-1: 2
Berat ikan ke-2: 3.5
Berat ikan ke-3: 5.78
Berat ikan ke-4: 1
Berat ikan ke-5: 1.5
Berat ikan ke-6: 3

Total berat ikan disetiap wadah :
Wadah ke-1 : 12.28 Kg
Sisanya di wadah ke-2 : 4.50 Kg

Rata-rata berat ikan disetiap wadah :
Wadah ke-1 : 3.07 Kg
Wadah ke-2 : 2.25 Kg
PS E:\alpro 2\src>
```

## Deskripsi Program

Program diatas digunakan untuk untuk menghitung total berat dan rata-rata berat ikan yang dimasukkan ke dalam beberapa wadah. Pengguna diminta memberikan input berupa jumlah total ikan, jumlah ikan per wadah, dan berat masing-masing ikan. Data berat ikan disimpan dalam array bertipe beratIkan. Fungsi totalBeratIkan menghitung dan mencetak total berat ikan di setiap wadah, sedangkan fungsi beratRataIkan menghitung dan mencetak rata-rata berat ikan di setiap wadah. Fungsi utama (main) mengoordinasikan input data dari pengguna, memprosesnya menggunakan kedua fungsi tersebut, dan menampilkan hasil total dan rata-rata berat ikan untuk setiap wadah.

## Sourcecode

```
package main

import "fmt"

type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, bMin, bMax *float64, n
int) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
```

```

        for i := 1; i < n; i++ {
            if *bMin > arrBerat[i] {
                *bMin = arrBerat[i]
            }
            if *bMax < arrBerat[i] {
                *bMax = arrBerat[i]
            }
        }
    }

func rerata(arrBerat arrBalita, n int) float64 {
    var rata float64 = 0
    bagi := 0
    for i := 0; i < n; i++ {
        rata += arrBerat[i]
        bagi++
    }

    return rata / float64(bagi)
}

func main() {
    var bMin, bMax float64
    var arrBerat arrBalita
    var n int

    fmt.Print("Masukkan banyak data berat balita : ")
    fmt.Scan(&n)

    if n < 1 || n > 100 {
        fmt.Println("Jumlah data berat balita harus antara
1 dan 100.")
        return
    }

    for i := 0; i < n; i++ {
        fmt.Print("Masukkan berat balita ke-", i+1, ": ")
        fmt.Scan(&arrBerat[i])
    }

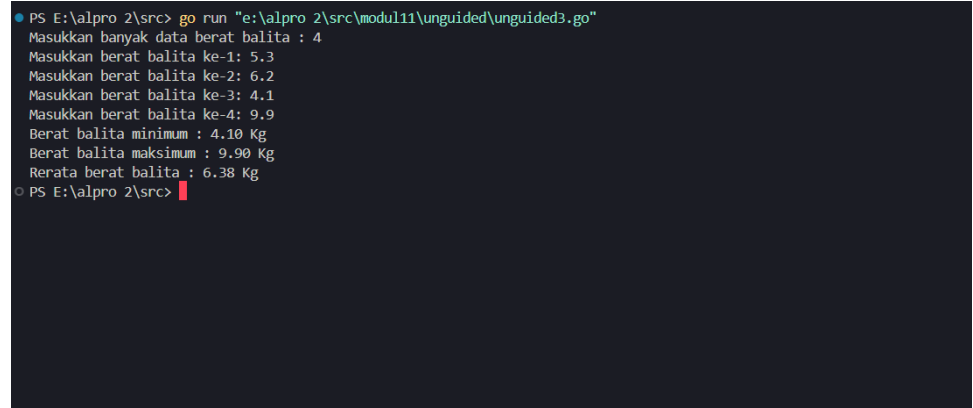
    hitungMinMax(arrBerat, &bMin, &bMax, n)
    rataRata := rerata(arrBerat, n)

```

```
fmt.Printf("Berat balita minimum : %.2f Kg\n", bMin)
fmt.Printf("Berat balita maksimum : %.2f Kg\n", bMax)
fmt.Printf("Rerata berat balita : %.2f Kg\n", rataRata)

}
```

### Screenshoot Output



```
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul11\unguided\unguided3.go"
Masukkan banyak data berat balita : 4
Masukkan berat balita ke-1: 5.3
Masukkan berat balita ke-2: 6.2
Masukkan berat balita ke-3: 4.1
Masukkan berat balita ke-4: 9.9
Berat balita minimum : 4.10 Kg
Berat balita maksimum : 9.90 Kg
Rerata berat balita : 6.38 Kg
PS E:\alpro 2\src>
```

### Deskripsi Program

*Program diatas digunakan untuk menghitung berat minimum, maksimum, dan rata-rata dari sekumpulan data berat balita. Pengguna diminta memasukkan jumlah data berat balita, lalu memasukkan nilai berat masing-masing balita satu per satu. Program menggunakan array bertipe `arrBalita` untuk menyimpan data berat dalam bentuk bilangan real. Fungsi `hitungMinMax` mencari nilai minimum dan maksimum berat balita serta menyimpannya ke variabel yang dirujuk oleh pointer. Fungsi `rerata` menghitung dan mengembalikan nilai rata-rata dari data berat balita. Fungsi utama (`main`) mengelola input data, memprosesnya dengan memanggil fungsi `hitungMinMax` dan `rerata`, lalu menampilkan hasil perhitungan berupa nilai berat minimum, maksimum, dan rata-rata.*