

**LAPORAN PRAKTIKUM PEMROGRAMAN  
BERORIENTASI OBJEK**

**MODUL 10**

**PENCARIAN NILAI EXSTRIM PADA HIMPUNAN DATA**



Oleh:

ARVAN MURBIYANTO

2311102174

IF - 11 – 02

**S1 TEKNIK INFORMATIKA TELKOM  
UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

### 1. Ide Pencarian Nilai Max/Min

Pencarian merupakan proses yang sering dilakukan dalam kehidupan sehari-hari. Contohnya, mencari file di direktori komputer, teks dalam dokumen, buku di rak, dan lainnya. Dalam modul ini, akan dibahas algoritma untuk mencari nilai terkecil atau terbesar dalam sekumpulan data, yang dikenal sebagai pencarian nilai ekstrem. Ide algoritmanya sangat sederhana. Data diproses secara berurutan, dan nilai atau indeks maksimum dari data yang telah diproses akan disimpan untuk dibandingkan dengan data berikutnya. Nilai terakhir yang tersimpan saat algoritma selesai adalah nilai maksimum yang dicari. Secara umum, algoritma ini melibatkan langkah-langkah berikut:

- 1) Menetapkan data pertama sebagai nilai ekstrem.
- 2) Memvalidasi nilai ekstrem dari data kedua hingga data terakhir.
  - Jika nilai ekstrem tidak valid, perbarui dengan data yang sedang diperiksa.
- 3) Setelah semua data diperiksa, nilai ekstrem yang tersimpan adalah hasil yang valid.

Berikut ini adalah notasi dalam pseudocode dan bahasa Go, misalnya untuk pencarian nilai terbesar atau maksimum:

	Notasi Algoritma	Notasi dalam bahasa Go
1	$\text{max} \leftarrow 1$	$\text{max} = 0$
2	$i \leftarrow 2$	$i = 1$
3	while $i \leq n$ do	for $i < n$ {
4	if $a[i] > a[\text{max}]$ then	if $a[i] > a[\text{max}]$ {
5	$\text{max} \leftarrow i$	$\text{max} = i$
6	endif	}
7	$i \leftarrow i + 1$	$i = i + 1$
8	endwhile	}

## 2. Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut. Perhatikan potongan program dalam bahasa Go berikut ini!

```
5  type arrInt [2023]int
..  ...
15
16 func terkecil_1(tabInt arrInt, n int) int {
17     /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
18     bilangan bulat */
19     var min int = tabInt[0]           // min berisi data pertama
20     var j int = 1                    // pencarian dimulai dari data berikutnya
21     for j < n {
22         if min > tabInt[j] {          // pengecekan apakah nilai minimum valid
23             min = tabInt[j]          // update nilai minimum dengan yang valid
24         }
25         j = j + 1
26     }
27     return min                       // returnkan nilai minimumnya
28 }
```

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada bahasa Go di mulai dari nol atau "0" seperti penjelasan pada modul 9. Selanjutnya, pada penjelasan di awal bab 3 telah disampaikan bahwa pada pencarian yang terpenting adalah posisi atau Indeks dari nilai yang dicari dalam kumpulan data atau array. Oleh karena itu modifikasi pada program di atas dapat dilihat pada potongan program berikut ini!

```
..  ...
5  type arrInt [2023]int
..  ...
15
16 func terkecil_2(tabInt arrInt, n int) int {
17     /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi
18     n bilangan bulat */
19     var idx int = 0                  // idx berisi indeks data pertama
20     var j int = 1                    // pencarian dimulai dari data berikutnya
21     for j < n {
22         if tabInt[idx] > tabInt[j] { // pengecekan apakah nilai minimum valid
23             idx = j                  // update nilai minimum dengan yang valid
24         }
25         j = j + 1
26     }
27     return idx                       // returnkan indeks nilai minimumnya
28 }
```

### 3. Pencarian Nilai Ekstrim pada Array Bertipe Data Terstruktur.

Pada kasus yang lebih kompleks pencarian ekstrim dapat juga dilakukan, misalnya mencari data mahasiswa dengan nilai terbesar, mencari lagu dengan durasi terlama, mencari pembalap yang memiliki catatan waktu balap tercepat, dan sebagainya. Sebagai contoh misalnya terdapat array yang digunakan untuk menyimpan data mahasiswa, kemudian terdapat fungsi IPK yang digunakan untuk mencari data mahasiswa dengan IPK tertinggi.

```
.. ...
5  type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
.. ...
15
16 func IPK_1(T arrMhs, n int) float64 {
17     /* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
18     n mahasiswa */
19     var tertinggi float64 = T[0].ipk
20     var j int = 1
21     for j < n {
22         if tertinggi < T[j].ipk {
23             tertinggi = T[j].ipk
24         }
25         j = j + 1
26     }
27     return tertinggi
28 }
```

Apabila diperhatikan potongan program di atas, maka kita akan memperoleh ipk tertinggi, tetapi kita tidak memperoleh identitas mahasiswa dengan ipk tertinggi tersebut. Maka seperti penjelasan yang sudah diberikan sebelumnya, maka pencarian yang dilakukan bisa mengembalikan indeks mahasiswa dengan ipk tertinggi tersebut.

## II. GUIDED

### Guided 1

```
package main

import "fmt"

func main() {
    var N int
    var berat [1000]float64

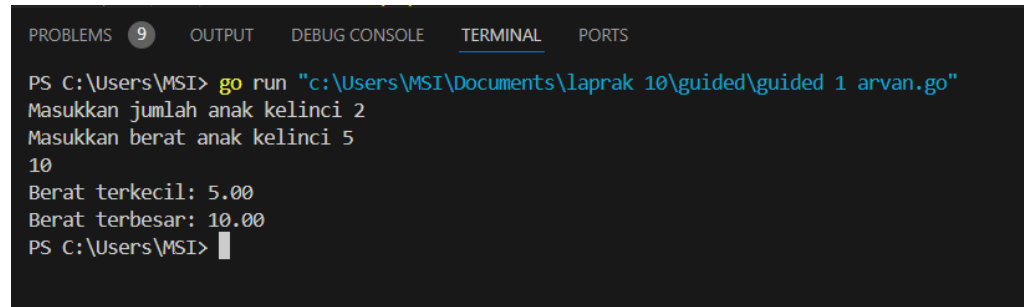
    fmt.Print("Masukkan jumlah anak kelinci ")
    fmt.Scan(&N)
    fmt.Print("Masukkan berat anak kelinci ")
    for i := 0; i < N; i++ {
        fmt.Scan(&berat[i])
    }

    min := berat[0]
    max := berat[0]

    for i := 0; i < N; i++ {
        if berat[i] < min {
            min = berat[i]
        }
        if berat[i] > max {
            max = berat[i]
        }
    }

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

## Screenshot output :

A screenshot of a Go IDE terminal window. The terminal shows the execution of a Go program. The user runs the command 'go run "c:\Users\MSI\Documents\laprak 10\guided\guided 1 arvan.go"'. The program prompts for the number of children ('Masukkan jumlah anak kelinci 2') and the weight of each child ('Masukkan berat anak kelinci 5'). The user enters '10'. The program then outputs 'Berat terkecil: 5.00' and 'Berat terbesar: 10.00'. The terminal window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active.

```
PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak 10\guided\guided 1 arvan.go"
Masukkan jumlah anak kelinci 2
Masukkan berat anak kelinci 5
10
Berat terkecil: 5.00
Berat terbesar: 10.00
PS C:\Users\MSI>
```

## Deskripsi program

Program di atas menentukan berat terkecil dan terbesar dari anak-anak kelinci. Setelah pengguna memasukkan jumlah dan berat tiap anak kelinci, program membandingkan setiap berat untuk menemukan nilai terkecil dan terbesar, lalu menampilkannya dalam dua desimal.

## Guided 2

```
package main

import (
    "fmt"
)

func main() {
    var x, y int
    fmt.Println("Masukkan jumlah ikan
dan kapasitas wadah: ")
    fmt.Scan(&x, &y)

    berat := make([]float64, x)
    fmt.Println("Masukkan berat tiap
ikan: ")
    for i := 0; i < x; i++ {
        fmt.Scan(&berat[i])
    }

    jumlahWadah := (x + y - 1) / y
    //pembulatan ke atas jika x tidak habis
dibagi y
    totalBeratWadah :=
make([]float64, jumlahWadah)

    for i := 0; i < x; i++ {
        indeksWadah := i / y
```

```

totalBeratWadah[indeksWadah] += berat[i]
    }

    //Output total berat tiap wadah
    fmt.Println("Total berat tiap wadah: ")
    for _, total := range totalBeratWadah {
        fmt.Printf("%.2f ", total)
    }
    fmt.Println()

    //Output rata-rata berat tiap wadah
    fmt.Println("Rata-rata berat tiap wadah: ")
    for _, total := range totalBeratWadah {
        rataRata := total / float64(y)
        fmt.Printf("%.2f ", rataRata)
    }
    fmt.Println()
}

```

## Screenshot output

```

PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak 10\guided\guided 2 arvan.go"
Masukkan jumlah ikan dan kapasitas wadah: 10 10
Masukkan berat tiap ikan:
2
2
3
4
3
6
5
2
4
5
Total berat tiap wadah:
36.00
Rata-rata berat tiap wadah:
3.60
PS C:\Users\MSI>

```

## Deskripsi program

Program ini menghitung total dan rata-rata berat ikan di setiap wadah berdasarkan jumlah ikan dan kapasitas wadah yang dimasukkan pengguna. Program mendistribusikan berat ikan ke wadah, kemudian menampilkan total berat dan rata-rata berat per wadah dengan dua angka desimal.

### III. UNGUIDED

#### Soal Modul 10

#### Source code no 3

```
package main

import "fmt"

const maxData = 100

type datas [maxData] float64

func minMax(data datas, n int) {
    min:= data[0]
    for i:=1; i < n; i++ {
        if min > data[i] {
            min = data[i]
        }
    }

    max := data[0]
    for i:=1; i < n; i++ {
        if max < data[i] {
            max = data[i]
        }
    }

    fmt.Printf("berat balita minimum: %.2f\n",min)
    fmt.Printf("berat balita maximum: %.2f\n",max)
}

func rataRata(data datas, n int) float64 {
    var sum,hasil float64

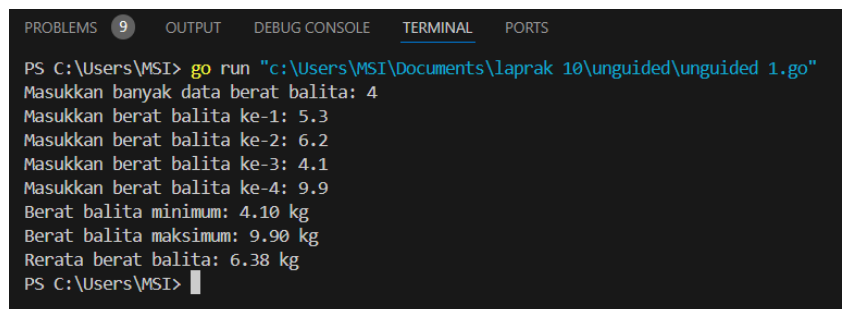
    for i:=0; i < n; i++ {
        sum += data[i]
    }
    hasil = float64(sum) / float64(n)

    return hasil
}
```



```
func main() {  
    var data datas  
    var n int  
  
    fmt.Scan(&n)  
  
    for i:=0; i < n; i++ {  
        fmt.Scan(&data[i])  
    }  
  
    minMax(data,n)  
    fmt.Printf("%.2f",rataRata(data,n))  
}
```

## Screenshot output



```
PROBLEMS 9 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\MSI> go run "c:\Users\MSI\Documents\laprak 10\unguided\unguided 1.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 C:\Users\MSI>
```

## Deskripsi

Program ini menghitung berat balita minimum, maksimum, dan rata-rata dari sekumpulan data. Pengguna memasukkan jumlah data dan berat balita, lalu program menggunakan fungsi minMax untuk mencari berat terkecil dan terbesar, serta fungsi rataRata untuk menghitung rata-rata berat balita. Hasilnya ditampilkan dengan format dua angka desimal.