

**LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL X
PENCARIAN NILAI EKSTRIM PADA
HIMPUNAN DATA**



Oleh:

NAMA : AHMAD TITANA NANDA PRAMUDYA

NIM : 2311102042

KELAS : IF 11 02

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
2024**

I. DASAR TEORI

Ide Pencarian Nilai Max/Min

Pencarian adalah suatu proses yang lazim dilakukan di dalam kehidupan sehari-hari. Contoh penggunaannya dalam kehidupan nyata sangat beragam, misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh lainnya. Pertama pada modul ini akan dipelajari salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data, atau biasa disebut pencarian nilai ekstrim.

Ide algoritma sederhana sekali. Karena data harus diproses secara sekuensial, maka nilai atau indeks ke nilai maksimum dari data yang telah diproses disimpan untuk dibandingkan dengan data berikutnya. Nilai yang berhasil disimpan sampai algoritma tersebut berakhir adalah nilai maksimum yang dicari. Adapun algoritmanya secara umum adalah sebagai berikut:

- 1) Jadikan data pertama sebagai nilai ekstrim
- 2) Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir. Apabila nilai ekstrim tidak valid, maka update nilai ekstrims tersebut dengan data yang dicek.
- 3) Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid matics lab

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

	Notasi Algoritma	Notasi Dalam Bahasa Go
	<pre>Max ← 1 i ← 2 while i ≤ n do if a[i] > a[max] then max ← i endif i ← i+1 endwhile</pre>	<pre>Max=0 i=1 for I < n { if a[i] > a[max] { max = i } i ← i+1 }</pre>

Pencarian Nilai Ekstrem pada Array Ber tipe 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

```
type arrInt [2023]int

func terkecil_1(tabInt arrInt, n int) int (
    /* inengeinballkan nilai terkecil yang terdapat di dalam tabInt yang berisi n
    bilangan bulat */
    var min int = tabInt[0] // inIn bert st data pertama
    var j int = 1 // pencarian diInula1 dari data berikutnya
    for j < n {
        if min » tabInt[j] { // pengecekan apakah nilai minInum valid
            min = tabInt[j] // update nilai inInInuin dengan yang valid
        }
        j = j + 1
    }
    return min // returnkan nilai minimumnya
}
```

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!

```
type arrInt [2023]int

func terkecil_2(tabInt arrInt, n int) int (
    /* nengeinballkan Indeks nilai terkecil yang terdapat di dalam tabInt yang berisi
    n bilangan bulat */
    var idx int = 0 // idx bert st Indeks data pertama
    var j int = 1 // pencarian diInula1 dari data berikutnya
    for j < n {
        if tabInt[idx] » tabInt[j] { // pengecekan apakah nilai minInum valid
            idx = j // update nilai inInInuin dengan yang valid
        }
        j = j + 1
    }
    return idx // returnkan indeks nilai minimumnya
}
```

Pencarian Nilai Ekstrem pada Array Berdasarkan Struktur

Pada kasus yang lebih kompleks pencarian ekstrem 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.

```
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk float64
}

type arrMhs [2023]mahasiswa

func IPK_1(T arrMhs, n int) float64 {
    /* mengembalikan Ipk terbesar yang dimiliki mahasiswa pada array T yang berisi
    n mahasiswa */
    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
}
```

II. GUIDED

No 1.

Source code:

```
package main

import (
    "fmt"
)

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

    fmt.Print("masukkan jumlah anak kelinci:")
    fmt.Scan(&N)

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

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

    for i := 1; 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)
}
```

Output :

```
PS D:\titan\titan 2\PRAKTIKUM ALPRO\2311102042_Ahmad Titana Nanda Pramudya_Modul10\Guided> go run guided1.go
masukkan jumlah anak kelinci:3
masukkan berat anak kelinci:
1
2
4
Berat terkecil: 1.00
Berat terbesar: 4.00
PS D:\titan\titan 2\PRAKTIKUM ALPRO\2311102042_Ahmad Titana Nanda Pramudya_Modul10\Guided> 
```

Penjelasan :

untuk menerima input berat beberapa anak kelinci, kemudian menentukan dan menampilkan berat terkecil dan terbesar dari data tersebut. Program ini bekerja dengan cara meminta jumlah anak kelinci, lalu mengumpulkan berat masing-masing anak kelinci ke dalam array berat. Setelah semua data dimasukkan, program membandingkan tiap nilai dalam array untuk menemukan nilai minimum dan maksimum, yang kemudian dicetak sebagai hasil akhir.

NO.2

Sourcode :

```
package main
import (
    "fmt"
)

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

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

    jumlahWadah := (x + y - 1) / y
    totalBeratWadah := make([]float64, jumlahWadah)

    for i := 0; i < x; i++ {
        indekswadah := i / y
        totalBeratWadah[indekswadah] += berat[i]
    }
    fmt.Println("total berat tiap wadah: ")
    for _, total := range totalBeratWadah {
        fmt.Printf("%.2f", total)
    }
    fmt.Println()

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

Output :

```
PS D:\titan\titan 2\PRAKTIKUM ALPRO\2311102042_Ahmad Titana Nanda Pramudya_Modul10\Guided> go run guided2.go
masukkan jumlah ikan dan kapasitas wadah: 1 3
masukkan berat tiap ikan: 12
total berat tiap wadah:
12.00
rata rata berat tiap wadah:
4.00
PS D:\titan\titan 2\PRAKTIKUM ALPRO\2311102042_Ahmad Titana Nanda Pramudya_Modul10\Guided> |
```

Penjelasan :

mengelompokkan ikan berdasarkan berat ke dalam beberapa wadah dengan kapasitas tertentu. Program ini meminta input dari pengguna terkait jumlah ikan, kapasitas wadah, dan berat masing-masing ikan. Setelah data dimasukkan, program menghitung jumlah wadah yang diperlukan dan mengalokasikan ikan ke dalam wadah sesuai kapasitas yang diberikan. Hasil akhirnya menampilkan total berat ikan di setiap wadah serta rata-rata berat per wadah, memberikan informasi ringkas tentang distribusi berat ikan tersebut.

III. UNGUIDE

Source code :

```
package main

import "fmt"

const maxData = 100

type datas [maxData]float64

func minMax(data datas, n int) (float64, float64) {
    min := data[0]
    max := data[0]

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

    return min, max
}

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

    for i := 0; i < n; i++ {
        sum += data[i]
    }

    return sum / float64(n)
}

func main() {
    var data datas
    var n int

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

    for i := 0; i < n; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ", i+1)
        fmt.Scan(&data[i])
    }
}
```

```
min, max := minMax(data, n)
fmt.Printf("\nBerat balita minimum: %.2f kg\n", min)
fmt.Printf("Berat balita maksimum: %.2f kg\n", max)
fmt.Printf("Rerata berat balita: %.2f kg\n", rataRata(data, n))
}
```

Output :

```
PS D:\titan\titan 2\PRAKTIKUM ALPRO\2311102042_Ahmad Titana Nanda Pramudya_Modul10\Unguided> go run unguided1.go
Masukan banyak data berat balita: 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan 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 D:\titan\titan 2\PRAKTIKUM ALPRO\2311102042_Ahmad Titana Nanda Pramudya_Modul10\Unguided> |
```

Penjelasan :

badan balita, lalu menghitung berat minimum, maksimum, dan rata-rata dari data yang dimasukkan oleh pengguna. Pengguna diminta untuk memasukkan jumlah balita dan berat badan masing-masing, dan program menggunakan dua fungsi utama, minMax dan rataRata. Hasil perhitungan berat minimum, maksimum, dan rata-rata ditampilkan dalam form. Program ini berguna untuk analisis berat badan balita dasar karena desain yang modular dan efisien. Selain itu, dapat dikembangkan lebih lanjut untuk tujuan lain, seperti memantau kesehatan atau gizi kelompok balita.