

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 8**

**PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



**Disusun Oleh :**

**Afif Rijal Azzami / 2311102235**

**IF-11-05**

**Dosen Pengampu :**

**Arif Amrulloh, S.Kom., M.Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. DASAR TEORI**

### **A. 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.

### **B. Pencarian Nilai Ekstrim Pada Array Bertipe Data Dasar.**

Array bertipe data dasar berisi elemen-elemen dengan tipe data sederhana seperti integer, float, char, atau string.

### **C. Pencarian Nilai Ekstrim Pada Array Bertipe Data Terstruktur.**

Array bertipe data terstruktur berisi elemen-elemen berupa tipe data kompleks seperti objek atau struct, yang memiliki beberapa atribut.

## II. UNGUIDED 1

### Sourcecode

```
package main

import (
    "fmt"
    "math"
)

// Fungsi untuk meminta input berat kelinci
func inputBeratKelinci(N235 int) []float64 {
    berat235 := make([]float64, N235)

    fmt.Println("Masukkan berat anak kelinci satu per satu:")

    for i := 0; i < N235; i++ {
        fmt.Printf("Berat kelinci ke-%d: ", i+1)

        fmt.Scan(&berat235[i])
    }

    return berat235
}

// Fungsi untuk mencari berat terkecil dan terbesar
func cariBeratTerbesarTerkecil(berat235 []float64) (float64, float64) {
    minBerat235 := math.MaxFloat64
    maxBerat235 := -math.MaxFloat64

    for _, b235 := range berat235 {
        if b235 < minBerat235 {
            minBerat235 = b235
        }
    }
}
```

```

    }

    if b235 > maxBerat235 {
        maxBerat235 = b235
    }
}

return minBerat235, maxBerat235
}

func main() {
    var N235 int

    fmt.Print("Masukkan jumlah anak kelinci: ")

    fmt.Scan(&N235)

    // Validasi jumlah kelinci

    if N235 <= 0 || N235 > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")
        return
    }

    // Input berat kelinci

    berat235 := inputBeratKelinci(N235)

    // Cari berat terkecil dan terbesar

    minBerat235, maxBerat235 :=
cariBeratTerbesarTerkecil(berat235)

```

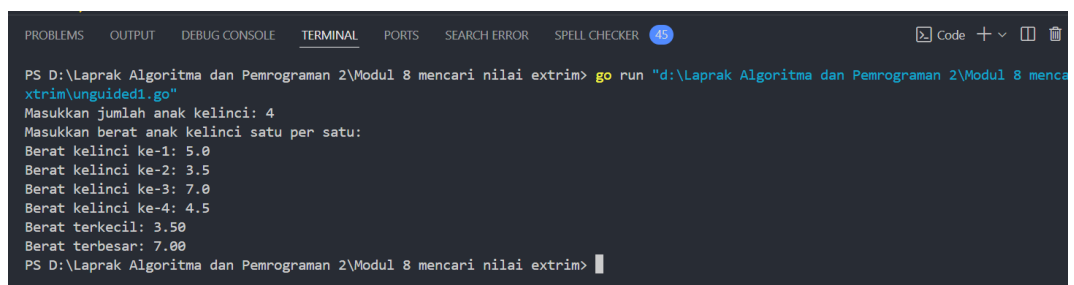
```
// Output hasil

fmt.Printf("Berat terkecil: %.2f\n", minBerat235)

fmt.Printf("Berat terbesar: %.2f\n", maxBerat235)

}
```

## Screenshoot Output



```
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 8 menca
xtrim\unguided1.go"
Masukkan jumlah anak kelinci: 4
Masukkan berat anak kelinci satu per satu:
Berat kelinci ke-1: 5.0
Berat kelinci ke-2: 3.5
Berat kelinci ke-3: 7.0
Berat kelinci ke-4: 4.5
Berat terkecil: 3.50
Berat terbesar: 7.00
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim> |
```

## Soal Studi Case

Program untuk mencari nilai max dan min pada array.

### Deskripsi Program

Sintaks tersebut adalah program untuk mencari berat max dan min pada berat kelinci yang diinputkan oleh user. Pertama kita membuat fungsi yang dinamai `inputBeratKelinci` untuk menginput berat kelinci yang nantinya disimpan pada array yang dinamai `berat` dengan panjang array yang ditentukan oleh user. Selanjutnya kita membuat fungsi yang dinamai `cariBeratTerbesarTerkecil` untuk mencari berat terkecil dan terbesar, cara kerjanya yaitu `minBerat` diinisialisasi dengan nilai maksimum untuk memastikan bahwa berat pertama dari array akan menjadi nilai awal terkecil, dan `maxBerat` diinisialisasi dengan nilai minimum untuk memastikan bahwa berat pertama dari array akan menjadi nilai awal terbesar. Selanjutnya terdapat pengondisian `if` jika nilai `b` lebih kecil dari pada `minBerat` maka `minBerat` diperbarui dengan nilai elemen tersebut. Dan jika `b` lebih besar dari `maxBerat` maka `maxBerat` diperbarui dengan nilai elemen tersebut.

### III. GUIDED 2

```
package main

import (
    "fmt"
    "math"
)

// Fungsi untuk membaca berat ikan dari input pengguna
func bacaBeratIkan235(jumlahIkan235 int) []float64 {
    beratIkan235 := make([]float64, jumlahIkan235)

    fmt.Println("Masukkan berat masing-masing ikan:")

    for i := 0; i < jumlahIkan235; i++ {
        fmt.Scan(&beratIkan235[i])
    }

    return beratIkan235
}

// Fungsi untuk mendistribusikan ikan ke dalam wadah
func distribusikanIkan235(beratIkan235 []float64,
    kapasitasWadah235 int) [][]float64 {
    jumlahWadah235 := int(math.Ceil(float64(len(beratIkan235)) /
    float64(kapasitasWadah235)))

    wadah235 := make([][]float64, jumlahWadah235)

    for i, berat235 := range beratIkan235 {
        indeksWadah235 := i / kapasitasWadah235

        wadah235[indeksWadah235] =
        append(wadah235[indeksWadah235], berat235)
    }
}
```

```

for i, berat235 := range beratIkan235 {

    indeksWadah235 := i / kapasitasWadah235

    wadah235[indeksWadah235] =
append(wadah235[indeksWadah235], berat235)

}

return wadah235
}

// Fungsi untuk menghitung total berat ikan di setiap wadah
func hitungTotalBerat235(wadah235 [][]float64) []float64 {

    totalBerat235 := make([]float64, len(wadah235))

    for i, isiWadah235 := range wadah235 {

        var jumlah235 float64

        for _, berat235 := range isiWadah235 {

            jumlah235 += berat235

        }

        totalBerat235[i] = jumlah235

    }

    return totalBerat235
}

// Fungsi untuk menghitung rata-rata berat ikan di setiap wadah
func hitungRataRataBerat235(wadah235 [][]float64) []float64 {

    rataRataBerat235 := make([]float64, len(wadah235))

    for i, isiWadah235 := range wadah235 {

        var jumlah235 float64

        for _, berat235 := range isiWadah235 {

            jumlah235 += berat235

        }

    }

```

```

}

    rataRataBerat235[i] = jumlah235 /
float64(len(isiWadah235))

}

    return rataRataBerat235
}

// Fungsi untuk menampilkan hasil total dan rata-rata berat
func tampilkanHasil235(totalBerat235, rataRataBerat235
[]float64) {

    fmt.Println("\nTotal berat ikan di setiap wadah:")

    for i, total235 := range totalBerat235 {

        fmt.Printf("Wadah %d: %.2f\n", i+1, total235)

    }

    fmt.Println("\nRata-rata berat ikan di setiap wadah:")

    for i, rataRata235 := range rataRataBerat235 {

        fmt.Printf("Wadah %d: %.2f\n", i+1, rataRata235)

    }

}

func main() {

    var jumlahIkan235, kapasitasWadah235 int

    fmt.Println("Masukkan jumlah ikan yang akan dijual (x) dan
kapasitas wadah (y):")

    fmt.Scan(&jumlahIkan235, &kapasitasWadah235)

```



```

// Memanggil fungsi untuk membaca input dan memproses data

beratIkan235 := bacaBeratIkan235(jumlahIkan235)

wadah235 := distribusikanIkan235(beratIkan235,
kapasitasWadah235)

totalBerat235 := hitungTotalBerat235(wadah235)

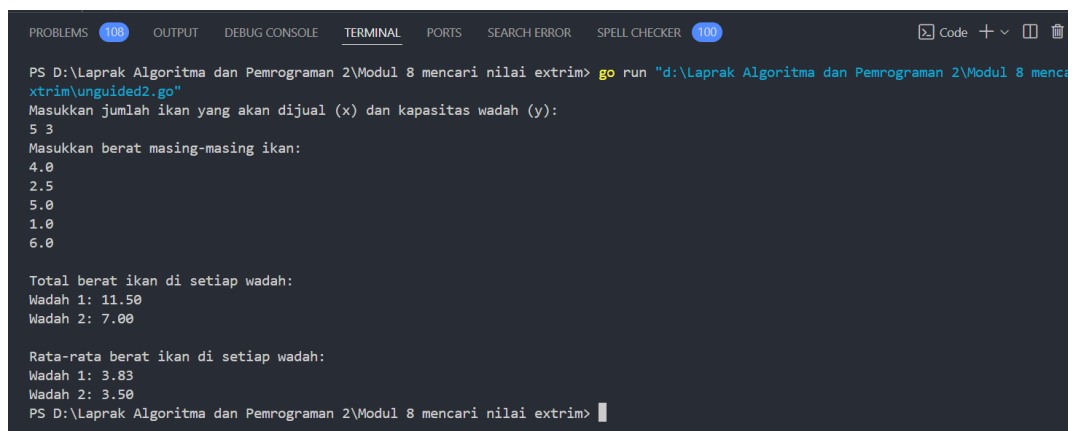
rataRataBerat235 := hitungRataRataBerat235(wadah235)

// Menampilkan hasil

tampilkanHasil235(totalBerat235, rataRataBerat235)
}

```

## Screenshoot Output



```

PS D:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim\unguided2.go"
Masukkan jumlah ikan yang akan dijual (x) dan kapasitas wadah (y):
5 3
Masukkan berat masing-masing ikan:
4.0
2.5
5.0
1.0
6.0

Total berat ikan di setiap wadah:
Wadah 1: 11.50
Wadah 2: 7.00

Rata-rata berat ikan di setiap wadah:
Wadah 1: 3.83
Wadah 2: 3.50
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim>

```

## Soal Studi Case

Mengelola distribusi berat ikan yang ingin dijual.

### Deskripsi Program

Sintaks tersebut adalah program untuk menampilkan berat ikan yang mau dijual dan mencari rata-rata pada setiap wadah yang berisi berat ikan yang mau dijual. Pertama kita membuat fungsi `bacaBeratIkan` untuk menyimpan berat ikan dalam array berat ikan dengan panjang arraynya jumlahIkan yang user tentukan. Selanjutnya kita membuat fungsi `distribusikan` untuk mendistribusikan berat ikan ke dalam wadah sesuai kapasitasnya, dan kita membuat fungsi `hitungTotalBerat` untuk menghitung total berat ikan di setiap

wadah, pertama kita membuat array totalBerat untuk menyimpan total berat ikan, dan fungsi tersebut melakukan iterasi yang dimana Menjumlahkan berat ikan di dalam wadah ke-i. selanjutnya kita membuat fungsi hitungRataBerat untuk menghitung rata-rata berat ikan pada setiap wadah, cara kerja fungsi tersebut adalah menjumlahkan berat ikan pada wadah dan selanjutnya menghitung rata-rata berat dengan membagi total berat dengan jumlah ikan, lalu hasil dari pembagian disimpan ke variabel rataRataBerat. Dan untuk fungsi tampilkan hasil untuk menampilkan hasil berat ikan dan hasil dari rata-rata berat yang sudah dihitung pada fungsi sebelumnya.

#### IV. UNGUIDED 3

```
package main

import "fmt"

// Tipe data array untuk menyimpan berat balita
type arrBalita235 [100]float64

// Fungsi untuk menghitung berat min dan max
func hitungMinMax235(arrBerat235 arrBalita235, n235 int)
(minBerat235, maxBerat235 float64) {

    if n235 == 0 {

        return 0, 0

    }

    minBerat235, maxBerat235 = arrBerat235[0], arrBerat235[0]

    for i235 := 1; i235 < n235; i235++ {

        if arrBerat235[i235] < minBerat235 {

            minBerat235 = arrBerat235[i235]

        }

        if arrBerat235[i235] > maxBerat235 {

            maxBerat235 = arrBerat235[i235]

        }

    }

    return

}
```

```
// Fungsi untuk menghitung rata-rata berat balita

func hitungRerata235(arrBerat235 arrBalita235, n235 int)
float64 {

    if n235 == 0 {

        return 0

    }

    var total235 float64

    for i235 := 0; i235 < n235; i235++ {

        total235 += arrBerat235[i235]

    }

    return total235 / float64(n235)
}

func main() {

    var n235 int

    var dataBerat235 arrBalita235

    for i235 := 0; i235 < len(dataBerat235); i235++ {

        dataBerat235[i235] = 0

    }

    // Input jumlah balita yang ingin di data

    fmt.Print("Banyak data berat balita : ")

    fmt.Scan(&n235)

    // mevalidasi jumlah data

    if n235 <= 0 || n235 > 100 {

        fmt.Println("Jumlah data tidak lebih dari 100.")

        return

    }

}
```

```
// Input data berat balita

for i235 := 0; i235 < n235; i235++ {

    fmt.Printf("Berat balita ke - %d : ", i235+1)

    fmt.Scan(&dataBerat235[i235])

}

// untuk menghitung minimum, maksimum, dan rata-rata berat
minBerat235, maxBerat235 := hitungMinMax235(dataBerat235,
n235)

rataRata235 := hitungRerata235(dataBerat235, n235)

// Output akhir

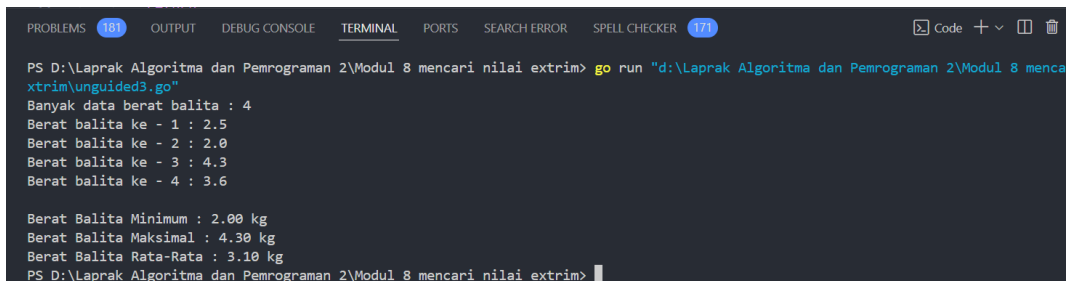
fmt.Printf("\nBerat Balita Minimum : %.2f kg\n",
minBerat235)

fmt.Printf("Berat Balita Maksimal : %.2f kg\n", maxBerat235)

fmt.Printf("Berat Balita Rata-Rata : %.2f kg\n",
rataRata235)

}
```

## Screenshot Output



```
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim\unguided3.go"
Banyak data berat balita : 4
Berat balita ke - 1 : 2.5
Berat balita ke - 2 : 2.0
Berat balita ke - 3 : 4.3
Berat balita ke - 4 : 3.6

Berat Balita Minimum : 2.00 kg
Berat Balita Maksimal : 4.30 kg
Berat Balita Rata-Rata : 3.10 kg
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 8 mencari nilai extrim>
```

## **Soal Studi Case**

Mencari nilai minimal, maksimal, dan rata-rata pada berat beberapa balita

## **Deskripsi Program**

Sintaks tersebut adalah program dengan untuk menghitung dan menampilkan berat minimum, maksimum, serta rata-rata balita yang diinput. Pertama kita membuat array bernama arrBalita yang memiliki kapasitas 100. Selanjutnya kita membuat fungsi hitungMinMax untuk mencari berat minimal ataupun maksimal pada balita, cara kerjanya yaitu jika elemen saat ini lebih kecil dari minBerat, maka nilai minBerat diperbarui dan jika elemen saat ini lebih besar dari maxBerat, maka nilai maxBerat diperbarui menjadi elemen tersebut. Selanjutnya kita membuat fungsi hitungRerata untuk mencari nilai rata-rata dari berat balita, cara kerjanya yaitu pada setiap iterasi, nilai elemen array arrBerat[i] ditambahkan ke variabel total, kemudian setelah semua elemen dijumlahkan, rata-rata dihitung dengan membagi total dengan jumlah elemen n.