# LAPORAN PRAKTIKUM
# ALGORITMA DAN PEMROGRAMAN 2

## MODUL 7

## PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA


Oleh:

RAJA MUHAMMAD LUFHTI

103112400027

IF-12-01


**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2025**

# I. DASAR TEORI

Dalam dunia pemrograman, pencarian nilai ekstrim (nilai minimum dan maksimum) dalam suatu kumpulan data merupakan proses yang sangat umum, terutama saat bekerja dengan array. Array adalah struktur data yang menyimpan sekumpulan nilai dalam satu variabel berdasarkan indeks. Dengan memanfaatkan array, kita dapat menyimpan data berat balita secara berurutan sesuai urutan input.

Proses pencarian nilai ekstrim dilakukan dengan membandingkan setiap elemen dalam array dengan nilai saat ini yang dianggap sebagai minimum atau maksimum. Awalnya, nilai pertama dalam array dijadikan acuan, lalu dibandingkan satu per satu dengan nilai lainnya. Jika ditemukan nilai yang lebih kecil atau lebih besar, nilai ekstrim diperbarui.

Selain itu, menghitung **rata-rata** merupakan proses umum lainnya dalam analisis data. Rata-rata dihitung dengan menjumlahkan seluruh elemen dalam array kemudian dibagi dengan jumlah data. Dalam konteks program ini, semua proses tersebut dibagi ke dalam subprogram atau fungsi agar lebih modular dan mudah dipelihara.

## II. GUIDED

1. Guided1

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

Output:

```
PS D:\ALPRO 2\103112400027_MODUL 7> go run "d:\ALPRO 2\103112400027_MODUL 7\guided1\1.go"
masukkan jumlah tanaman: 3
masukkan tinggi tanaman ke-1 (cm): 12
masukkan tinggi tanaman ke-2 (cm): 20
masukkan tinggi tanaman ke-3 (cm): 15
Tinggi tanaman terpendek: 12.00 cm
Tinggi tanaman tertinggi: 20.00 cm
PS D:\ALPRO 2\103112400027_MODUL 7>
```

Deskripsi program:

Program ini ditulis dalam bahasa go dan berfungsi untuk mengumpulkan dan menganalisis data tinggi tanaman. Pertama program meminta inputan dari pengguna untuk menginputkan jumlah tanaman.

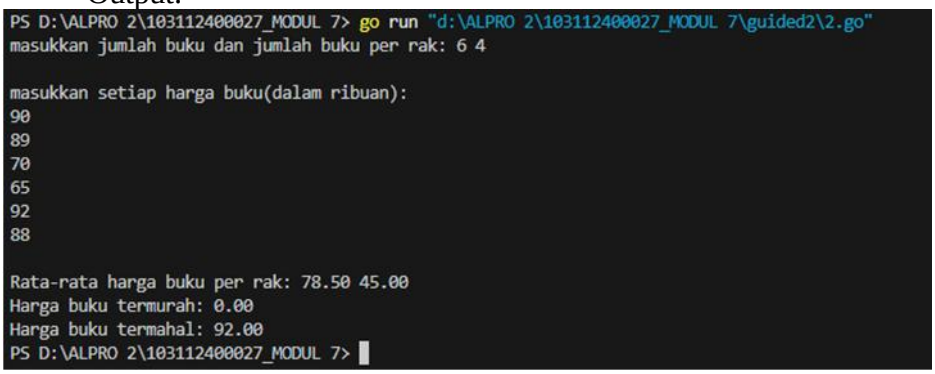
Setelah jumlah tanaman ditentukan, program akan menginisialisasi sebuah array dengan kapasitas maksimum 500 elemen yang bertipe float64, digunakan untuk menyimpan tinggi tanaman dalam satuan centi meter. Selanjutnya, pengguna akan diminta memasukkan tinggi masing-masing tanaman satu per satu, dan nilai-nilai tersebut akan disimpan di dalam array. Setelah data semua tinggi tanaman diinput program melanjutkan untuk mencari mana tanaman yang tertinggi dan terpendek. Dengan loop program melakukan perbandingan antar setiap elemen dalam array untuk menemukan nilai minimum dan maksimum. Setelah proses pencarian selesai, program mencetak tinggi tanaman tertinggi dan terpendek, menampilkan hasilnya dengan dua angka di belakang koma.

2. Guided2

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 setiap harga buku(dalam
ribuan):")
    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 {
        if harga < min {
            min = harga
        }
        if harga > max {
            max = harga
        }
    }

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

Output:




Deskripsi program:

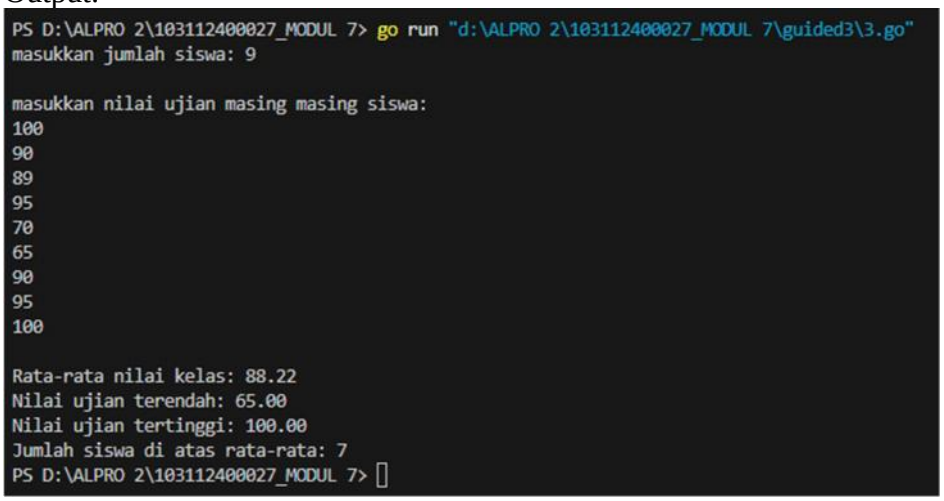
Program diatas ditulis dalam bahasa go dan berfungsi untuk mengelola dan menganalisis harga buku yang disimpan dalam rak. Pertama program meminta pengguna untuk memasukkan dua nilai yaitu jumlah total buku yang akan diinput dan jumlah buku yang dapat disimpan per rak. Setelah itu, program menginisialisasi sebuah array dengan kapasitas maksimum 500 elemen bertipe float64 untuk menyimpan harga setiap buku dalam satuan ribuan Rupiah. Pengguna kemudian diminta untuk memasukkan harga setiap buku secara bertahap, yang disimpan dalam array hargaBuku. Setelah semua harga buku dimasukkan, program akan menghitung rata-rata harga buku per rak. Program akan menggunakan dua loop yaitu loop luar untuk iterasi setiap rak dan loop dalam untuk menjumlahkan harga buku yang ada di dalam satu rak. Rata-rata harga untuk masing-masing rak kemudian disimpan dalam slice hargaRataRata. Selanjutnya, program akan mencari harga buku termahal dan termurah dengan membandingkan setiap elemen dalam array hargaBuku. Setelah proses pencarian selesai, program akan mencetak rata-rata harga per rak, serta harga termahal dan termurah dan menampilkan hasilnya dengan dua angka di belakang koma.

3. Guided3

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("\nRata-rata nilai kelas: %.2f", rataRata)
    fmt.Printf("\nNilai ujian terendah: %.2f", min)
    fmt.Printf("\nNilai ujian tertinggi: %.2f", max)
    fmt.Printf("\nJumlah siswa di atas rata-rata: %d\n", diAtasRataRata)

}
```

Output:




Deskripsi program:

Program diatas ditulis dalam bahasa go dan berfungsi untuk mengelola dan menganalisis nilai ujian siswa. Pertama program meminta pengguna untuk menginputkan jumlah siswa. Setelah diinputkan program menginisialisasi sebuah array dengan kapasitas maksimum 200 elemen bertipe float64 untuk menyimpan nilai ujian masing-masing siswa. Program kemudian meminta pengguna untuk menginputkan nilai ujian satu per satu, dan secara bersamaan menghitung total nilai yang dimasukkan. Setelah semua nilai siswa diinput, program akan menghitung rata-rata nilai kelas dengan cara membagi total nilai dengan jumlah siswa. Selanjutnya, program mencari nilai terendah dan tertinggi di antara nilai-nilai yang telah diinput. Selain itu, program juga menghitung jumlah siswa yang memiliki nilai di atas rata-rata. Untuk melakukan ini, program menggunakan loop untuk membandingkan setiap nilai dengan nilai minimum, maksimum, dan rata-rata yang telah dihitung sebelumnya. Setelah semua analisis selesai, program akan mencetak hasil, termasuk nilai terendah, nilai tertinggi, rata-rata kelas, dan jumlah siswa yang memiliki nilai di atas rata-rata.

## III. UNGUIDED

1. Unguided1

Source code:

```
//RAJA MUHAMMAD LUFHTI_103112400027
package main

import "fmt"

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

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

    fmt.Println("Masukkan berat anak kelinci:")
    for i := 0; i < jumlahAnak; i++ {
        fmt.Printf("Berat anak kelinci ke-%d: ", i+1)
        fmt.Scan(&berat[i])
    }

    beratTerkecil := berat[0]
    beratTerbesar := berat[0]

    for i := 1; i < jumlahAnak; i++ {
        if berat[i] < beratTerkecil {
            beratTerkecil = berat[i]
        }
        if berat[i] > beratTerbesar {
            beratTerbesar = berat[i]
        }
    }

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

Output:

```
PS D:\ALPRO 2\103112400027_MODUL 7> go run "d:\ALPRO 2\103112400027_MODUL 7\un1\u1.go"
Masukkan jumlah anak kelinci: 4
Masukkan berat anak kelinci:
Berat anak kelinci ke-1: 8
Berat anak kelinci ke-2: 5
Berat anak kelinci ke-3: 4.5
Berat anak kelinci ke-4: 6
Berat terkecil: 4.50
Berat terbesar: 8.00
PS D:\ALPRO 2\103112400027_MODUL 7>
```

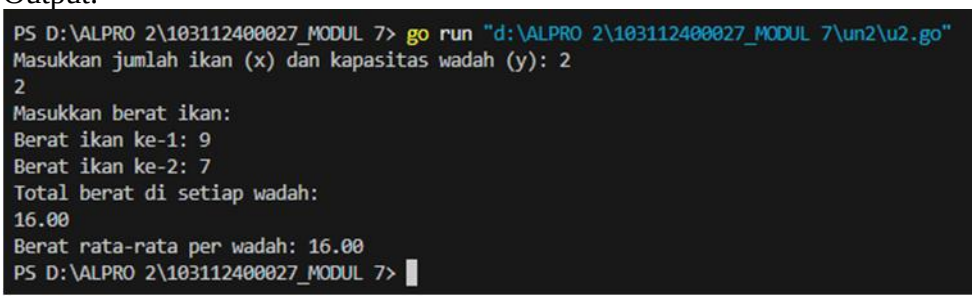
Deskripsi program:
Program diatas ditulis dalam bahasa go dan berfungsi mencatat serta menghitung berat anak kelinci yang akan dijual. Di dalam fungsi main, program mendeklarasikan variabel N untuk menyimpan jumlah anak kelinci dan sebuah array weights dengan kapasitas 1000 untuk menyimpan berat masing-masing kelinci dalam tipe data float64. Kemudian program meminta pengguna untuk menginputkan jumlah anak kelinci. Setelah pengguna menginputkan jumlah tersebut, program melanjutkan dengan meminta pengguna untuk menginputkan berat setiap anak kelinci satu per satu. Dalam loop for, program mencetak pesan yang meminta berat anak kelinci ke-i dan menyimpan nilai yang dimasukkan ke dalam array weights. Setelah semua berat diinput, program menginisialisasi dua variabel, minWeight dan maxWeight, dengan nilai berat anak kelinci pertama. Kemudian, program melakukan iterasi melalui sisa elemen dalam array weights untuk mencari nilai berat terkecil dan terbesar. Jika berat kelinci yang sedang diperiksa lebih kecil dari minWeight maka minWeight akan diperbarui dengan nilai tersebut. Sebaliknya, jika berat kelinci lebih besar dari maxWeight maka maxWeight juga akan diperbarui. Setelah proses pencarian selesai, program mencetak hasilnya dengan menampilkan berat terkecil dan terbesar yang ditemukan, masing-masing dengan format dua angka dibelakang koma.

2. Unguided2

Source code:

```
//RAJA MUHAMMAD LUFHTI_103112400027
package main

import (
    "fmt"
)

func main() {
    var x, y int
    var ikan [1000]float64

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

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

    var totalWadah [1000]float64
    jumlahWadah := (x + y - 1) / y

    for i := 0; i < x; i++ {
        indexWadah := i / y
        totalWadah[indexWadah] += ikan[i]
    }

    fmt.Print("Total berat di setiap wadah:\n")
    for i := 0; i < jumlahWadah; i++ {
        fmt.Printf("%.2f ", totalWadah[i])
    }
    fmt.Println()

    var total float64
    for i := 0; i < jumlahWadah; i++ {
        total += totalWadah[i]
    }
    rataRata := total / float64(jumlahWadah)
    fmt.Printf("Berat rata-rata per wadah: %.2f\n",
rataRata)
}
```

Output:




Deskripsi program:

Program diatas ditulis dalam bahasa go dan berfungsi menghitung total berat ikan yang dimasukkan ke dalam wadah berdasarkan kapasitas yang ditentukan. Program dimulai dengan mendefinisikan fungsi tarif yang menerima dua parameter yaitu sebuah slice bertipe float64 yang berisi berat ikan dan sebuah integer y yang menunjukkan kapasitas maksimum ikan yang dapat dimuat dalam satu wadah. Dalam fungsi tarif ada dua variabel utama yaitu variabel total yang merupakan slice untuk menyimpan total berat ikan per wadah dan variabel jumlah yang digunakan untuk menghitung total berat ikan dalam wadah saat ini. Kemudian program melakukan perulangan melalui setiap berat ikan yang diberikan. Setiap kali berat ikan ditambahkan ke jumlah, program memeriksa apakah jumlah ikan dalam wadah telah mencapai kapasitas y atau jika sudah mencapai ikan terakhir. Jika salah satu kondisi tersebut terpenuhi, total berat ikan dalam wadah saat ini ditambahkan ke slice total, dan variabel jumlah serta count direset untuk memulai perhitungan wadah berikutnya. Selanjutnya program akan menghitung rata-rata berat ikan per wadah dengan menjumlahkan semua total berat yang ada di slice total dan membaginya dengan jumlah wadah yang telah dibuat. Fungsi tarif kemudian mengembalikan slice total dan nilai rata-rata tersebut. Pada fungsi main program meminta pengguna untuk menginputkan jumlah ikan dan kapasitas wadah. Lalu pengguna diminta untuk menginputkan berat masing-masing ikan. Program kemudian memanggil fungsi tarif dengan data yang telah diinput dan mencetak total berat ikan per wadah serta rata-rata berat ikan di setiap wadah.

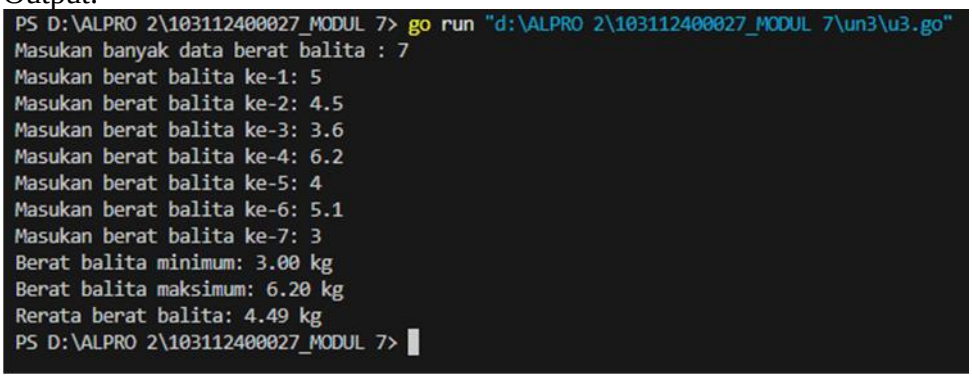
3. Unguided3

Source code:

```
//RAJA MUHAMMAD LUFHTI_103112400027
package main

import (
    "fmt"
)

type arrBalita [100]float64

func hitungMinMax(arr arrBalita, jumlah int, bMin *float64, bMax
*float64) {
    *bMin = arr[0]
    *bMax = arr[0]

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

func rerata(arr arrBalita, jumlah int) float64 {
    var total float64 = 0
    for i := 0; i < jumlah; i++ {
        total += arr[i]
    }
    return total / float64(jumlah)
}

func main() {
    var data arrBalita
    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])
    }

    var min, max float64
    hitungMinMax(data, n, &min, &max)
    rerataBerat := rerata(data, n)
```

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

Output:




Deskripsi program:

Program diatas ditulis dalam bahasa go dan berfungsi menghitung berat balita dengan beberapa statistik dasar, yaitu berat minimum, maksimum, dan rerata. Program dimulai dengan mendefinisikan sebuah tipe data baru bernama arrBalita yang merupakan array dengan kapasitas maksimum 100 elemen bertipe data float64. Tipe data ini digunakan untuk menyimpan berat balita yang diinput oleh pengguna. Fungsi hitungMinMax berfungsi untuk mencari nilai minimum dan maksimum dari array berat balita. Fungsi ini menerima tiga parameter yaitu arrBerat (array berat balita), n (jumlah data yang dimasukkan), dan dua pointer bMin dan bMax untuk menyimpan hasil minimum dan maksimum. Di fungsi ini nilai minimum dan maksimum diinisialisasi dengan elemen pertama dari array, kemudian dilakukan iterasi untuk membandingkan setiap elemen dengan nilai minimum dan maksimum yang ada, memperbarui nilai tersebut jika ditemukan elemen yang lebih kecil atau lebih besar. Fungsi rerata berguna menghitung rata-rata berat balita dengan menjumlahkan semua elemen dalam array dan membaginya dengan jumlah elemen yang ada. Fungsi ini menerima parameter arrBerat dan n, dan mengembalikan nilai rata-rata sebagai float64. Pada fungsi main program meminta pengguna untuk menginput jumlah data berat balita yang ingin dimasukkan. Kemudian program melakukan perulangan untuk meminta pengguna menginput berat balita satu per satu untuk disimpan dalam array berat. Setelah semua data diinput, program memanggil fungsi hitungMinMax untuk mendapatkan nilai minimum dan maksimum, serta fungsi rerata untuk menghitung rata-rata berat balita. Terakhir program akan mencetak hasil dari berat balita minimum, maksimum, dan rerata.

# IV. KESIMPULAN

Program pencatatan berat balita ini membuktikan pentingnya penggunaan array dalam menangani kumpulan data yang seragam dan berurutan. Dengan bantuan subprogram hitungMinMax dan rerata, program menjadi lebih terstruktur, efisien, dan mudah dikembangkan. Penggunaan array memungkinkan pengolahan data dalam jumlah banyak dengan cara yang sistematis.
Dari implementasi ini, kita dapat menyimpulkan bahwa pencarian nilai minimum, maksimum, dan perhitungan rata-rata dapat dilakukan secara sederhana namun efektif hanya dengan satu kali traversal array. Teknik ini penting dalam dunia nyata, seperti pengelolaan data kesehatan balita di posyandu, untuk pengambilan keputusan berbasis data.

# V. REFERENSI

MODUL 10.