

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2  
MODUL XI  
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



**Oleh:**

**M Zuljalali Ikram Noer(19102299)**

**IF-11-07**

**S1 TEKNIK INFORMATIKA  
UNIVERSITAS TELKOM PURWOKERTO  
2024**

# **I. DASAR TEORI**

## **1. 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.

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

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.

## **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.

## II. GUIDED

### Guided 1

#### Source Code

```
package main

import "fmt"

// Definisi tipe data array dengan ukuran tetap
type arrInt [2023]int

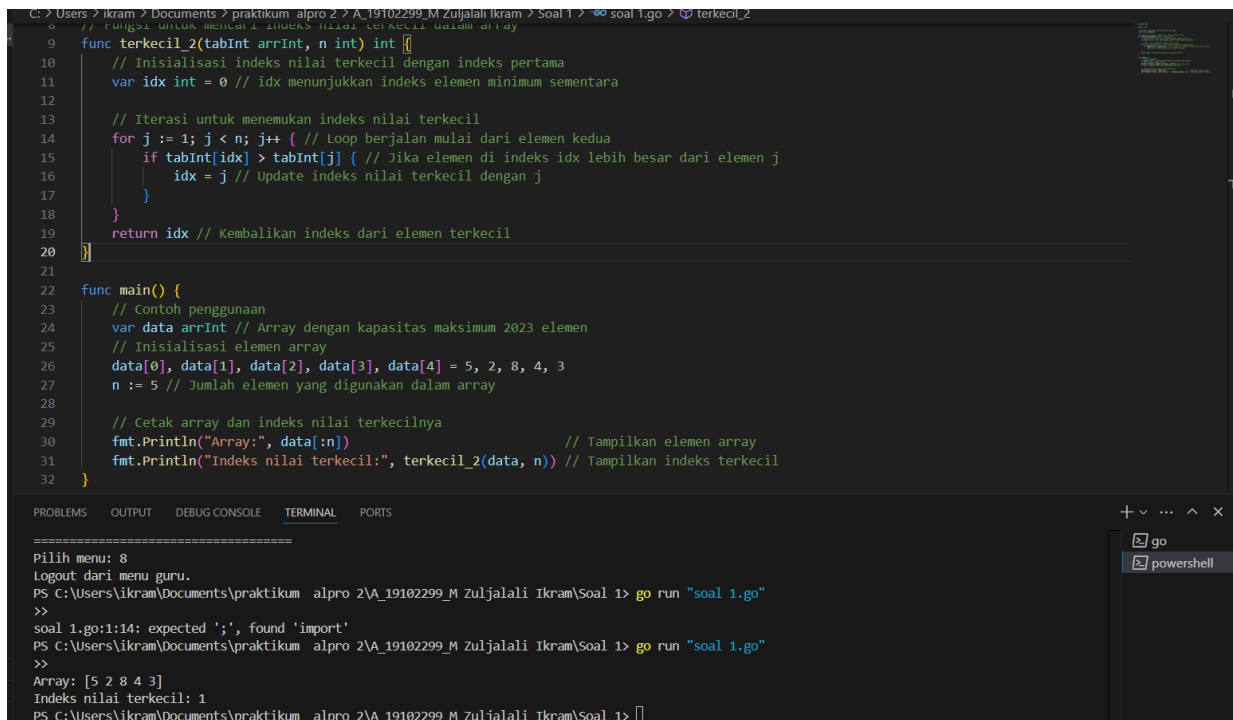
// Fungsi untuk mencari indeks nilai terkecil dalam array
func terkecil_2(tabInt arrInt, n int) int {
    // Inisialisasi indeks nilai terkecil dengan indeks pertama
    var idx int = 0 // idx menunjukkan indeks elemen minimum sementara

    // Iterasi untuk menemukan indeks nilai terkecil
    for j := 1; j < n; j++ { // Loop berjalan mulai dari elemen kedua
        if tabInt[idx] > tabInt[j] { // Jika elemen di indeks idx lebih
            // besar dari elemen j
            idx = j // Update indeks nilai terkecil dengan j
        }
    }
    return idx // Kembalikan indeks dari elemen terkecil
}

func main() {
    // Contoh penggunaan
    var data arrInt // Array dengan kapasitas maksimum 2023 elemen
    // Inisialisasi elemen array
    data[0], data[1], data[2], data[3], data[4] = 5, 2, 8, 4, 3
    n := 5 // Jumlah elemen yang digunakan dalam array

    // Cetak array dan indeks nilai terkecilnya
    fmt.Println("Array:", data[:n]) // Tampilkan elemen
    // array
    fmt.Println("Indeks nilai terkecil:", terkecil_2(data, n)) // Tampilkan
    // indeks terkecil
}
```

## Screenshot Program



```
C:\Users\ikram\Documents\praktikum_alpro 2\A_19102299_M Zuljalali Ikram\Soal 1> go run "soal 1.go"
9 func terkecil_2(tabInt arrInt, n int) int {
10     // Inisialisasi indeks nilai terkecil dengan indeks pertama
11     var idx int = 0 // idx menunjukkan indeks elemen minimum sementara
12
13     // Iterasi untuk menemukan indeks nilai terkecil
14     for j := 1; j < n; j++ { // Loop berjalan mulai dari elemen kedua
15         if tabInt[idx] > tabInt[j] { // Jika elemen di indeks idx lebih besar dari elemen j
16             idx = j // Update indeks nilai terkecil dengan j
17         }
18     }
19     return idx // Kembalikan indeks dari elemen terkecil
20 }
21
22 func main() {
23     // Contoh penggunaan
24     var data arrInt // Array dengan kapasitas maksimum 2023 elemen
25     // Inisialisasi elemen array
26     data[0], data[1], data[2], data[3], data[4] = 5, 2, 8, 4, 3
27     n := 5 // Jumlah elemen yang digunakan dalam array
28
29     // Cetak array dan indeks nilai terkecilnya
30     fmt.Println("Array:", data[:n]) // Tampilkan elemen array
31     fmt.Println("Indeks nilai terkecil:", terkecil_2(data, n)) // Tampilkan indeks terkecil
32 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
=====
Pilih menu: 8
Logout dari menu guru.
PS C:\Users\ikram\Documents\praktikum_alpro 2\A_19102299_M Zuljalali Ikram\Soal 1> go run "soal 1.go"
>>
soal 1.go:1:14: expected ';', found 'import'
PS C:\Users\ikram\Documents\praktikum_alpro 2\A_19102299_M Zuljalali Ikram\Soal 1> go run "soal 1.go"
>>
Array: [5 2 8 4 3]
Indeks nilai terkecil: 1
PS C:\Users\ikram\Documents\praktikum_alpro 2\A_19102299_M Zuljalali Ikram\Soal 1>
```

## Penjelasan

Program di atas mendefinisikan fungsi pencarian nilai terkecil dalam sebuah array tetap. Tipe data `arrInt` didefinisikan sebagai array dengan ukuran tetap 2023, meskipun pada implementasi, hanya lima elemen yang diinisialisasi. Fungsi `terkecil_2` menerima array `arrInt` dan jumlah elemen `n`, lalu mengembalikan indeks elemen dengan nilai terkecil. Dalam fungsi ini, iterasi dilakukan mulai dari indeks ke-1 hingga `n-1`, membandingkan setiap elemen dengan elemen pada indeks minimum sementara. Jika ditemukan elemen lebih kecil, indeks minimum diperbarui.

Pada fungsi `main`, array `data` diisi dengan lima nilai, yaitu 5, 2, 8, 4, 3, dan `n` diatur ke 5 untuk menunjukkan jumlah elemen yang digunakan. Fungsi `terkecil_2` kemudian dipanggil untuk mencari indeks elemen terkecil dalam array tersebut. Hasilnya dicetak, termasuk elemen-elemen array dan indeks dari elemen terkecil. Dalam contoh ini, nilai terkecil adalah 2, yang terletak di indeks 1.

## Guided 2

### Source code

```

package main

import "fmt"

// Definisi struktur untuk menyimpan data mahasiswa
type mahasiswa struct {
    nama, nim, kelas, jurusan string // Informasi mahasiswa: nama, nim,
    kelas, jurusan
    ipk float64 // Nilai IPK mahasiswa
}

// Definisi array untuk menampung data mahasiswa
type arrMhs [2023]mahasiswa // Array statis dengan maksimum 2023
mahasiswa

// Fungsi untuk mencari indeks mahasiswa dengan IPK tertinggi
func IPK_2(T arrMhs, n int) int {
    // Inisialisasi indeks nilai IPK tertinggi dengan elemen pertama
    var idx int = 0 // idx menunjukkan indeks mahasiswa dengan IPK
    tertinggi sementara

    // Iterasi untuk membandingkan IPK mahasiswa
    for j := 1; j < n; j++ { // Pencarian dimulai dari mahasiswa kedua
        if T[idx].ipk < T[j].ipk { // Jika IPK pada idx lebih kecil dari IPK
        pada indeks j
            idx = j // Update indeks mahasiswa dengan IPK tertinggi
        }
    }
    return idx // Kembalikan indeks mahasiswa dengan IPK tertinggi
}

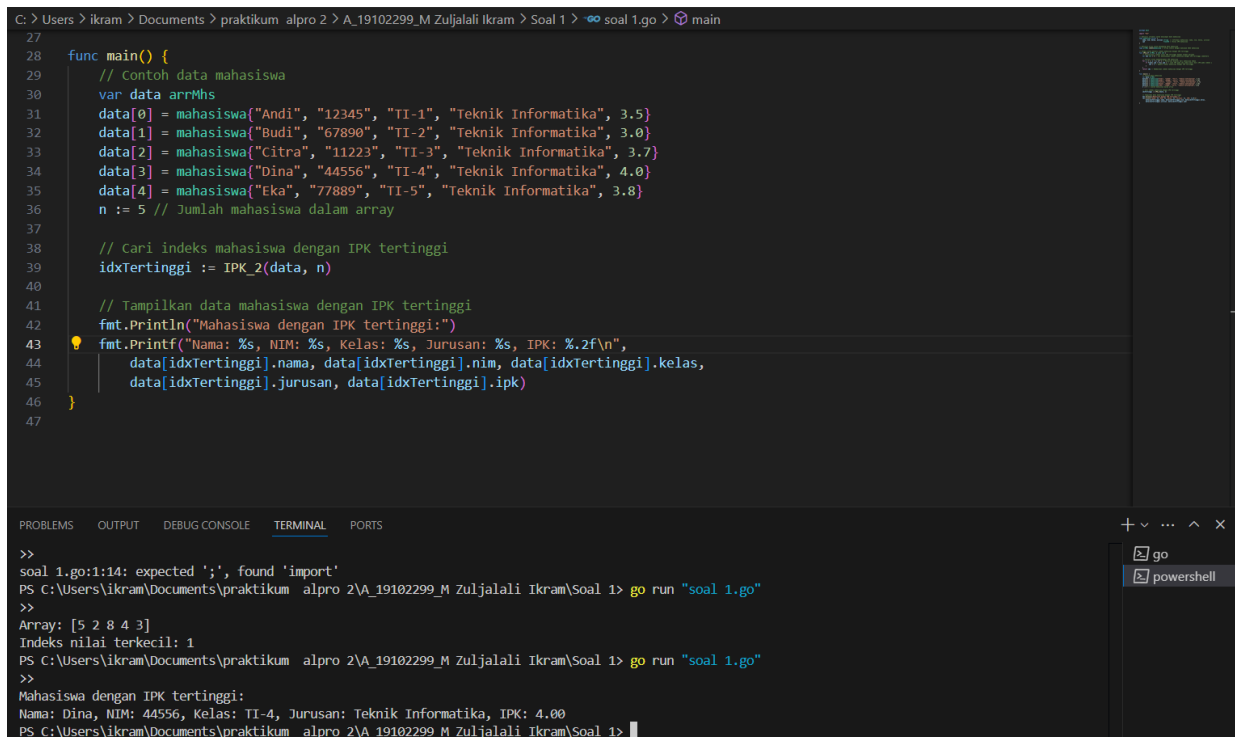
func main() {
    // Contoh data mahasiswa
    var data arrMhs
    data[0] = mahasiswa{"Andi", "12345", "TI-1", "Teknik Informatika", 3.5}
    data[1] = mahasiswa{"Budi", "67890", "TI-2", "Teknik Informatika", 3.0}
    data[2] = mahasiswa{"Citra", "11223", "TI-3", "Teknik Informatika",
3.7}
    data[3] = mahasiswa{"Dina", "44556", "TI-4", "Teknik Informatika", 4.0}
    data[4] = mahasiswa{"Eka", "77889", "TI-5", "Teknik Informatika", 3.8}
    n := 5 // Jumlah mahasiswa dalam array

    // Cari indeks mahasiswa dengan IPK tertinggi
    idxTertinggi := IPK_2(data, n)

    // Tampilkan data mahasiswa dengan IPK tertinggi
    fmt.Println("Mahasiswa dengan IPK tertinggi:")
    fmt.Printf("Nama: %s, NIM: %s, Kelas: %s, Jurusan: %s, IPK: %.2f\n",
        data[idxTertinggi].nama, data[idxTertinggi].nim,
    data[idxTertinggi].kelas,
        data[idxTertinggi].jurusan, data[idxTertinggi].ipk)
}

```

## Screenshot Program



```
27
28 func main() {
29     // Contoh data mahasiswa
30     var data arrMhs
31     data[0] = mahasiswa{"Andi", "12345", "TI-1", "Teknik Informatika", 3.5}
32     data[1] = mahasiswa{"Budi", "67890", "TI-2", "Teknik Informatika", 3.0}
33     data[2] = mahasiswa{"Citra", "11223", "TI-3", "Teknik Informatika", 3.7}
34     data[3] = mahasiswa{"Dina", "44556", "TI-4", "Teknik Informatika", 4.0}
35     data[4] = mahasiswa{"Eka", "77889", "TI-5", "Teknik Informatika", 3.8}
36     n := 5 // Jumlah mahasiswa dalam array
37
38     // Cari indeks mahasiswa dengan IPK tertinggi
39     idxTertinggi := IPK_2(data, n)
40
41     // Tampilkan data mahasiswa dengan IPK tertinggi
42     fmt.Println("Mahasiswa dengan IPK tertinggi:")
43     fmt.Printf("Nama: %s, NIM: %s, Kelas: %s, Jurusan: %s, IPK: %.2f\n",
44         data[idxTertinggi].nama, data[idxTertinggi].nim, data[idxTertinggi].kelas,
45         data[idxTertinggi].jurusan, data[idxTertinggi].ipk)
46 }
47
```

```
>>
soal 1.go:1:14: expected ';', found 'import'
PS C:\Users\ikram\Documents\praktikum alpro 2\A_19102299_M Zuljalali Ikram\Soal 1> go run "soal 1.go"
>>
Array: [5 2 8 4 3]
Indeks nilai terkecil: 1
PS C:\Users\ikram\Documents\praktikum alpro 2\A_19102299_M Zuljalali Ikram\Soal 1> go run "soal 1.go"
>>
Mahasiswa dengan IPK tertinggi:
Nama: Dina, NIM: 44556, Kelas: TI-4, Jurusan: Teknik Informatika, IPK: 4.00
PS C:\Users\ikram\Documents\praktikum alpro 2\A_19102299_M Zuljalali Ikram\Soal 1>
```

## Penjelasan

Program ini untuk memproses data mahasiswa yang disimpan dalam array. Struktur mahasiswa digunakan untuk menyimpan informasi seperti nama, NIM, kelas, jurusan, dan IPK. Tipe data `arrMhs` didefinisikan sebagai array statis berkapasitas maksimal 2023 elemen, yang menyimpan data berupa objek mahasiswa. Fungsi `IPK_2` digunakan untuk mencari indeks mahasiswa dengan IPK tertinggi dalam array. Fungsi ini membandingkan setiap elemen dari array berdasarkan IPK-nya, memperbarui indeks jika ditemukan mahasiswa dengan IPK lebih tinggi.

Pada fungsi `main`, array `data` diisi dengan lima data mahasiswa. Fungsi `IPK_2` dipanggil untuk menentukan indeks mahasiswa dengan IPK tertinggi, yang hasilnya adalah mahasiswa dengan IPK 4.0 atas nama Dina. Hasil ini dicetak ke layar dalam format yang menampilkan semua informasi mahasiswa tersebut. Program ini berguna untuk menyaring mahasiswa berprestasi berdasarkan nilai IPK tertinggi.

### III. UNGUIDED

#### Unguided 1

##### Source Code

```
package main

import "fmt"

type arrFloat [1000]float64

func beratTerkecil(data arrFloat, n int) float64 {
    min := data[0]
    for i := 1; i < n; i++ {
        if data[i] < min {
            min = data[i]
        }
    }
    return min
}

func beratTerbesar(data arrFloat, n int) float64 {
    max := data[0]
    for i := 1; i < n; i++ {
        if data[i] > max {
            max = data[i]
        }
    }
    return max
}

func main() {

    var data arrFloat
    var n int

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

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

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

```

    min := beratTerkecil(data, n)
    max := beratTerbesar(data, n)

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

```

## Screenshot Program

```

C:\> Users > ikram > Documents > praktikum alpro 2 > A_19102299_M Zuljalali Ikram > Soal 1 > go run "soal 1.go" > ...
30 func main() {
35 // Input jumlah anak kelinci
36 fmt.Print("Masukkan jumlah anak kelinci: ")
37 fmt.Scan(&n)
38
39 // Validasi jumlah anak kelinci
40 if n <= 0 || n > 1000 {
41     fmt.Println("Jumlah anak kelinci harus antara 1 hingga 1000.")
42     return
43 }
44
45 // Input berat setiap kelinci
46 for i := 0; i < n; i++ {
47     fmt.Printf("Masukkan berat kelinci ke-%d: ", i+1)
48     fmt.Scan(&data[i])
49 }
50
51 // Hitung berat terkecil dan terbesar
52 min := beratTerkecil(data, n)
53 max := beratTerbesar(data, n)
54
55 // Tampilkan hasil
56 fmt.Printf("Berat terkecil: %.2f\n", min)
57 fmt.Printf("Berat terbesar: %.2f\n", max)
58 }

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```

>>
Mahasiswa dengan IPK tertinggi:
Nama: Dina, NIM: 44556, Kelas: TI-4, Jurusan: Teknik Informatika, IPK: 4.00
PS C:\Users\ikram\Documents\praktikum alpro 2\A_19102299_M Zuljalali Ikram\Soal 1> go run "soal 1.go"
>>
Masukkan jumlah anak kelinci: 2
Masukkan berat kelinci ke-1: 3
Masukkan berat kelinci ke-2: 4
Berat terkecil: 3.00
Berat terbesar: 4.00
PS C:\Users\ikram\Documents\praktikum alpro 2\A_19102299_M Zuljalali Ikram\Soal 1>

```

## Penjelasan

Program ini menentukan berat terkecil dan terbesar dari sejumlah anak kelinci berdasarkan data yang diinputkan pengguna. Tipe data `arrFloat` adalah array statis dengan kapasitas maksimum 1000 elemen bertipe `float64`. Fungsi `beratTerkecil` dan `beratTerbesar` masing-masing digunakan untuk mencari nilai minimum dan maksimum dalam array. Keduanya menggunakan pendekatan iteratif dengan membandingkan setiap elemen terhadap nilai minimum atau maksimum sementara yang diperbarui selama iterasi.

Pada fungsi `main`, program meminta pengguna memasukkan jumlah anak kelinci (`n`), yang harus berada dalam rentang 1 hingga 1000. Jika jumlah tidak valid, program menghentikan eksekusi. Setelah itu, berat setiap kelinci dimasukkan ke array `data`. Fungsi `beratTerkecil` dan `beratTerbesar` kemudian dipanggil untuk menghitung berat minimum dan maksimum, dan hasilnya ditampilkan dengan format dua desimal. Program ini berguna untuk menganalisis data berat secara sederhana dan efisien.



## Unguided 2

### Source Code

```
package main

import "fmt"

type arrFloat [1000]float64

func totalBeratPerWadah(data arrFloat, x, y int) [1000]float64 {
    var wadah [1000]float64
    for i := 0; i < x; i++ {
        wadah[i%y] += data[i]
    }
    return wadah
}

func rataRataPerWadah(wadah [1000]float64, y int) float64 {
    var totalBerat float64
    var wadahCount int

    for i := 0; i < y; i++ {
        totalBerat += wadah[i]
        if wadah[i] > 0 {
            wadahCount++
        }
    }

    if wadahCount == 0 {
        return 0
    }
    return totalBerat / float64(wadahCount)
}

func main() {
    var data arrFloat
    var x, y int

    fmt.Print("Masukkan jumlah ikan yang akan dijual (x): ")
    fmt.Scan(&x)
    fmt.Print("Masukkan jumlah wadah (y): ")
    fmt.Scan(&y)
```

```

    if x <= 0 || x > 1000 || y <= 0 {
        fmt.Println("Input tidak valid. Pastikan jumlah ikan (x) antara
1 hingga 1000 dan jumlah wadah (y) lebih dari 0.")
        return
    }

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

    wadah := totalBeratPerWadah(data, x, y)

    fmt.Println("Total berat ikan di setiap wadah:")
    for i := 0; i < y; i++ {
        fmt.Printf("Wadah %d: %.2f\n", i+1, wadah[i])
    }

    average := rataRataPerWadah(wadah, y)

    fmt.Printf("Berat rata-rata ikan di setiap wadah: %.2f\n", average)
}

```

## Screenshot Program

```

C: > Users > ikram > Documents > praktikum_alpro 2 > A_19102299_M Zuljalali Ikram > Soal 1 > go soal1.go > ...
35 func main() {
52     for i := 0; i < x; i++ {
53         fmt.Printf("Masukkan berat ikan ke-%d: ", i+1)
54         fmt.Scan(&data[i])
55     }
56
57     // Menghitung total berat ikan per wadah
58     wadah := totalBeratPerWadah(data, x, y)
59
60     // Menampilkan total berat ikan di setiap wadah
61     fmt.Println("Total berat ikan di setiap wadah:")
62     for i := 0; i < y; i++ {
63         fmt.Printf("Wadah %d: %.2f\n", i+1, wadah[i])
64     }
65
66     // Menghitung dan menampilkan berat rata-rata ikan di setiap wadah
67     average := rataRataPerWadah(wadah, y)
68     fmt.Printf("Berat rata-rata ikan di setiap wadah: %.2f\n", average)
69 }
70
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan jumlah wadah (y): 4
Masukkan berat ikan ke-1: 5
Masukkan berat ikan ke-2: 5
Masukkan berat ikan ke-3: 4
Total berat ikan di setiap wadah:
Wadah 1: 5.00
Wadah 2: 5.00
Wadah 3: 4.00
Wadah 4: 0.00
Berat rata-rata ikan di setiap wadah: 4.67
PS C:\Users\ikram\Documents\praktikum_alpro 2\A_19102299_M Zuljalali Ikram\Soal 1>

```

## Penjelasan

Program ini untuk menghitung total berat ikan yang ditempatkan dalam beberapa wadah dan mencari rata-rata berat ikan per wadah. Array `arrFloat` digunakan untuk menyimpan data berat ikan, dengan kapasitas maksimum 1000 elemen. Fungsi `totalBeratPerWadah` menghitung total berat ikan di setiap wadah, dengan membagi ikan secara berurutan ke wadah-wadah menggunakan operasi modulus ( $i \% y$ ). Fungsi ini mengembalikan array berisi total berat ikan yang ada di masing-masing wadah.

Fungsi `rataRataPerWadah` menghitung rata-rata berat ikan per wadah dengan menjumlahkan berat ikan di semua wadah yang memiliki ikan, kemudian membagi total berat dengan jumlah wadah yang terisi. Dalam fungsi `main`, program meminta pengguna untuk memasukkan jumlah ikan ( $x$ ) dan jumlah wadah ( $y$ ), serta memastikan input valid. Setelah itu, berat ikan dimasukkan ke dalam array `data`. Fungsi `totalBeratPerWadah` dipanggil untuk menghitung total berat di masing-masing wadah, yang kemudian ditampilkan. Program juga menghitung dan mencetak rata-rata berat ikan per wadah, memberikan gambaran distribusi ikan yang lebih merata ke dalam wadah.

## Unguided 3

### Source Code

```
package main

import (
    "fmt"
)

type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, n int, bMin, bMax *float64) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]
    for i := 1; i < n; i++ {
        if arrBerat[i] < *bMin {
            *bMin = arrBerat[i]
        }
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
    }
}

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

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

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

```

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

    hitungMinMax(arrBerat, n, &bMin, &bMax)

    rerataBerat := 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", rerataBerat)
}

```

## Screenshot Program

```

C:\Users\ikram\Documents\praktikum_alpro 2 > A_19102299_M Zuljalali Ikram > Soal 1 > soal 1.go > ...
32 func main() {
46
47     // Input berat balita
48     for i := 0; i < n; i++ {
49         fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
50         fmt.Scan(&arrBerat[i])
51     }
52
53     // Menghitung berat minimum dan maksimum
54     hitungMinMax(arrBerat, n, &bMin, &bMax)
55
56     // Menghitung rerata berat
57     rerataBerat := rerata(arrBerat, n)
58
59     // Menampilkan hasil
60     fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
61     fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)
62     fmt.Printf("Rerata berat balita: %.2f kg\n", rerataBerat)
63 }
64
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Berat rata-rata ikan di setiap wadah: 4.67
PS C:\Users\ikram\Documents\praktikum_alpro 2\A_19102299_M Zuljalali Ikram\Soal 1> go run "soal 1.go"
>>
Masukkan banyak data berat balita: 3
Masukkan berat balita ke-1: 4
Masukkan berat balita ke-2: 3
Masukkan berat balita ke-3: 2
Berat balita minimum: 2.00 kg
Berat balita maksimum: 4.00 kg
Rerata berat balita: 3.00 kg
PS C:\Users\ikram\Documents\praktikum_alpro 2\A_19102299_M Zuljalali Ikram\Soal 1>
Ln 64, Col 1 Tab Size: 4 UTF-8 CRLF Go 1.23.2

```

## Penjelasan

Program ini untuk menganalisis data berat badan balita dengan menghitung nilai minimum, maksimum, dan rata-rata dari data yang dimasukkan oleh pengguna. Data berat disimpan dalam array `arrBalita` dengan kapasitas maksimum 100 elemen. Fungsi `hitungMinMax` digunakan untuk menemukan nilai minimum dan maksimum dengan membandingkan setiap elemen dalam array, sementara fungsi `rerata` menghitung rata-rata berat badan dengan menjumlahkan semua elemen dan membaginya dengan jumlah data. Proses ini menggunakan teknik iterasi untuk memastikan semua elemen dianalisis secara efisien.

Pada fungsi `main`, program meminta pengguna memasukkan jumlah data balita (`n`) dan berat masing-masing balita, yang kemudian disimpan dalam array. Setelah itu, fungsi `hitungMinMax` dan `rerata` dipanggil untuk menghitung berat minimum, maksimum, dan rata-rata. Hasil analisis ditampilkan dalam format yang informatif, menunjukkan berat terkecil, terbesar, dan rata-rata dalam dua angka desimal.