

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 11

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

Martin Christopher Simbolon 2311102269

Kelas IF-11-06

Dosen Pengampu :

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Dasar Teori

Pencarian nilai ekstrem, yaitu nilai maksimum dan minimum, pada suatu himpunan data merupakan salah satu konsep dasar dalam algoritma dan pemrograman (alpro). Hal ini sering menjadi bagian awal dalam pembelajaran pemrograman karena melibatkan penggunaan logika iterasi dan seleksi, yang merupakan inti dari banyak algoritma. Pencarian nilai ekstrem digunakan untuk menentukan batas atas dan bawah dalam data, yang berguna dalam analisis statistik, optimasi, dan pengambilan keputusan.

Secara sederhana, pencarian nilai ekstrem dilakukan dengan membandingkan elemen-elemen dalam himpunan data satu per satu. Dalam sebuah algoritma dasar, nilai awal (seed) untuk maksimum dan minimum biasanya diinisialisasi dengan elemen pertama dari data. Kemudian, melalui proses iterasi, setiap elemen dibandingkan dengan nilai maksimum dan minimum sementara, dan pembaruan dilakukan jika elemen tersebut lebih besar atau lebih kecil dari nilai ekstrem sementara. Algoritma ini memiliki kompleksitas waktu $O(n)$, di mana n adalah jumlah elemen dalam himpunan data.

II. GUIDED

1.Sourcecode

```
package main

import "fmt"

// Mendeklarasikan tipe data array arrInt dengan panjang 2023
type arrInt [2023]int

// Fungsi untuk mencari elemen terkecil dalam array
func terkecil(tabInt arrInt, n int) int {
    var min int = tabInt[0]
    var j int = 1
    for j < n {
        if min > tabInt[j] {
            min = tabInt[j]
        }
        j = j + 1
    }
    return min
}

// Fungsi main
func main() {
    var n int
    var tab arrInt

    // Meminta input jumlah elemen array
    fmt.Print("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n)

    // Validasi input jumlah elemen
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah elemen harus antara 1 dan 2023.")
        return
    }

    // Memasukkan elemen-elemen array
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
        fmt.Print("Elemen ke-", i+1, ": ")
        fmt.Scan(&tab[i])
    }
}
```

```

    }

    // Memanggil fungsi terkecil untuk menemukan nilai terkecil
    minVal := terkecil(tab, n)

    // Menampilkan nilai terkecil
    fmt.Println("Nilai terkecil dalam array adalah:", minVal)
}

```

Screenshoot Output

```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL SEARCH ERROR
Elemen ke-1: 4
Elemen ke-2: 5
Elemen ke-3: 6
Elemen ke-4: 7
Elemen ke-5: 8
Elemen ke-6: 3
Elemen ke-7: 2
Elemen ke-8: 5
Elemen ke-9: 3
Elemen ke-10: 1
Nilai terkecil dalam array adalah: 1
PS C:\Users\MARTIN SIMBOLON>

```

Deskripsi Program

Program ini bertujuan untuk mencari nilai terkecil dalam sebuah array yang berisi angka-angka yang dimasukkan oleh pengguna. Pada awalnya, program mendeklarasikan tipe data array `arrInt` dengan kapasitas maksimum 2023 elemen. Fungsi utama dalam program ini adalah `terkecil`, yang menerima array dan panjang elemen array sebagai parameter, kemudian mencari elemen terkecil dalam array tersebut dengan menggunakan iterasi.

2. Sourcecode

```

package main

import "fmt"

// Mendeklarasikan tipe data array arrInt dengan panjang 2023
type arrInt [2023]int

// Fungsi untuk mencari indeks elemen terkecil dalam array
func terkecil(tabInt arrInt, n int) int {
    var idx int = 0 // idx menyimpan indeks elemen terkecil
    var j int = 1

```

```

    for j < n {
        if tabInt[idx] > tabInt[j] {
            idx = j // Simpan indeks j jika elemen di indeks j lebih kecil
        }
        j = j + 1
    }
    return idx
}

// Fungsi main untuk menguji fungsi terkecil
func main() {
    var n int
    var tab arrInt

    // Meminta input jumlah elemen array
    fmt.Print("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n)

    // Validasi input jumlah elemen
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah elemen harus antara 1 dan 2023.")
        return
    }

    // Memasukkan elemen-elemen array
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
        fmt.Print("Elemen ke-", i+1, ": ")
        fmt.Scan(&tab[i])
    }

    // Memanggil fungsi terkecil untuk menemukan indeks elemen terkecil
    idxMin := terkecil(tab, n)

    // Menampilkan nilai dan indeks terkecil
    fmt.Println("Nilai terkecil dalam array adalah:", tab[idxMin], "pada indeks:",
idxMin)
}

```

Screenshoot output

```
ip.c8e\Modul 11\TipeDasarIDX.go"
Masukkan jumlah elemen (maks 2023): 5
Masukkan elemen-elemen array:
Elemen ke-1: 2
Elemen ke-2: 3
Elemen ke-3: 4
Elemen ke-4: 5
Elemen ke-5: 6
Nilai terkecil dalam array adalah: 2 pada indeks: 0
```

Deskripsi

Di dalam fungsi `terkecil`, program memulai pencarian dengan mengasumsikan bahwa elemen pertama (indeks 0) adalah yang terkecil, dan menyimpan indeksnya dalam variabel `idx`. Kemudian, program akan membandingkan elemen pertama dengan elemen-elemen lainnya dalam array. Jika ditemukan elemen yang lebih kecil dari nilai yang ada di indeks `idx`, maka indeks `idx` akan diperbarui dengan indeks elemen yang lebih kecil tersebut. Proses ini berlanjut hingga seluruh elemen array diperiksa. Setelah selesai, fungsi `terkecil` mengembalikan indeks dari elemen terkecil.

3. Sourcecode

```
package main

import "fmt"

// Definisi struct mahasiswa dengan atribut nama, nim, kelas,
// jurusan, dan ipk
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                        float64
}

// Definisi tipe data array mahasiswa dengan kapasitas maksimal
// 2023
type arrMhs [2023]mahasiswa

// Fungsi untuk mencari indeks IPK tertinggi dalam array
// mahasiswa
func indeksIPKTertinggi(T arrMhs, n int) int {
    var idx int = 0 // Inisialisasi indeks IPK tertinggi pada indeks
    pertama
    for j := 1; j < n; j++ {
        if T[idx].ipk < T[j].ipk {
            idx = j // Update indeks jika ditemukan IPK yang lebih
            tinggi
        }
    }
    return idx
}
```

```

    }
}
return idx
}

// Fungsi main untuk mengisi data mahasiswa dan mencari IPK tertinggi beserta indeks nya
func main() {
    var n int
    var dataMhs arrMhs

    // Meminta input jumlah mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")
    fmt.Scan(&n)

    // Validasi jumlah mahasiswa yang dimasukkan
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")
        return
    }

    // Mengisi data mahasiswa
    for i := 0; i < n; i++ {
        fmt.Printf("\nMasukkan data mahasiswa ke-%d\n", i+1)
        fmt.Print("Nama: ")
        fmt.Scan(&dataMhs[i].nama)
        fmt.Print("NIM: ")
        fmt.Scan(&dataMhs[i].nim)
        fmt.Print("Kelas: ")
        fmt.Scan(&dataMhs[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scan(&dataMhs[i].jurusan)
        fmt.Print("IPK: ")
        fmt.Scan(&dataMhs[i].ipk)
    }

    // Mendapatkan indeks IPK tertinggi
    idxTertinggi := indeksIPKTertinggi(dataMhs, n)

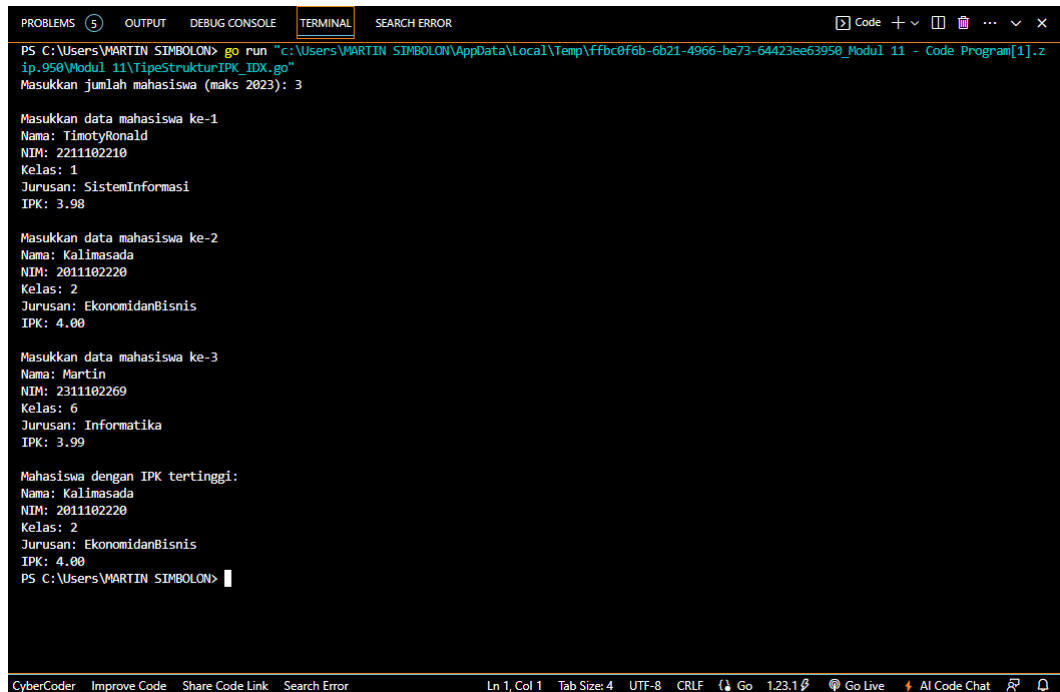
```

```

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

```

Screenshoot output



```

PS C:\Users\WARTIN SIMBOLON> go run "c:\Users\WARTIN SIMBOLON\AppData\Local\Temp\ffbc0f6b-6b21-4966-be73-64423ee63950_Modul 11 - Code Program[1].z
ip_950\Modul 11\TipeStrukturIPK_IDX.go"
Masukkan jumlah mahasiswa (maks 2023): 3

Masukkan data mahasiswa ke-1
Nama: TimotyRonald
NIM: 2211102210
Kelas: 1
Jurusan: SistemInformasi
IPK: 3.98

Masukkan data mahasiswa ke-2
Nama: Kalimasada
NIM: 2011102220
Kelas: 2
Jurusan: EkonomidanBisnis
IPK: 4.00

Masukkan data mahasiswa ke-3
Nama: Martin
NIM: 2311102269
Kelas: 6
Jurusan: Informatika
IPK: 3.99

Mahasiswa dengan IPK tertinggi:
Nama: Kalimasada
NIM: 2011102220
Kelas: 2
Jurusan: EkonomidanBisnis
IPK: 4.00
PS C:\Users\WARTIN SIMBOLON>

```

Deskripsi Program

Program di atas dirancang untuk mengelola data mahasiswa dan mencari mahasiswa dengan IPK tertinggi. Program ini menggunakan tipe data `struct` untuk menyimpan informasi tentang mahasiswa, yang terdiri dari nama, NIM, kelas, jurusan, dan IPK. Program meminta input dari pengguna untuk menentukan jumlah mahasiswa (maksimal 2023) dan kemudian mengisi data mahasiswa tersebut, termasuk nama, NIM, kelas, jurusan, dan IPK. Setelah semua data dimasukkan, program mencari indeks mahasiswa dengan IPK tertinggi menggunakan fungsi `indeksIPKTertinggi`, yang membandingkan nilai IPK setiap mahasiswa. Setelah menemukan mahasiswa dengan IPK tertinggi, program menampilkan informasi lengkap mahasiswa tersebut, termasuk nama, NIM, kelas, jurusan, dan nilai IPK tertinggi. Program ini memanfaatkan array untuk menyimpan data mahasiswa dan dapat digunakan untuk aplikasi manajemen data akademik sederhana.

4. Sourcecode

```
package main

import "fmt"

// Definisi struct mahasiswa dengan atribut nama, nim, kelas,
jurusan, dan ipk
type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                float64
}

// Definisi tipe data array mahasiswa dengan kapasitas maksimal
2023
type arrMhs [2023]mahasiswa

// Fungsi untuk mencari indeks IPK tertinggi dalam array
mahasiswa
func indeksIPKTertinggi(T arrMhs, n int) int {
    var idx int = 0 // Inisialisasi indeks IPK tertinggi pada indeks
pertama
    for j := 1; j < n; j++ {
        if T[idx].ipk < T[j].ipk {
            idx = j // Update indeks jika ditemukan IPK yang lebih
tinggi
        }
    }
    return idx
}

// Fungsi main untuk mengisi data mahasiswa dan mencari IPK
tertinggi beserta indeksnya
func main() {
    var n int
    var dataMhs arrMhs

    // Meminta input jumlah mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")
    fmt.Scan(&n)
```

```

// Validasi jumlah mahasiswa yang dimasukkan
if n < 1 || n > 2023 {
    fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")
    return
}

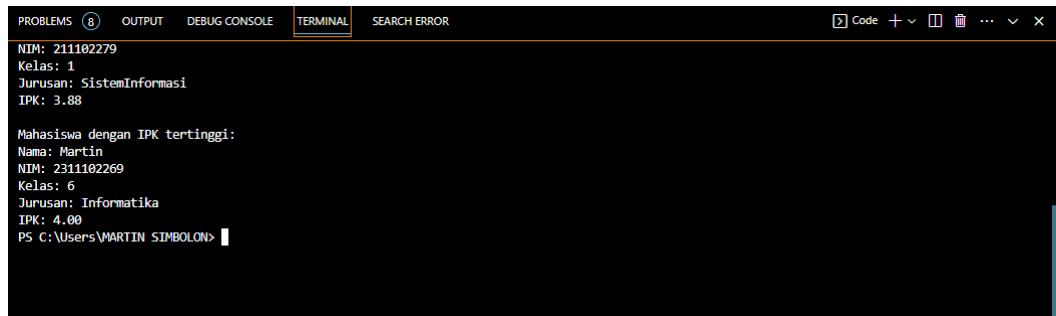
// Mengisi data mahasiswa
for i := 0; i < n; i++ {
    fmt.Printf("\nMasukkan data mahasiswa ke-%d\n", i+1)
    fmt.Print("Nama: ")
    fmt.Scan(&dataMhs[i].nama)
    fmt.Print("NIM: ")
    fmt.Scan(&dataMhs[i].nim)
    fmt.Print("Kelas: ")
    fmt.Scan(&dataMhs[i].kelas)
    fmt.Print("Jurusan: ")
    fmt.Scan(&dataMhs[i].jurusan)
    fmt.Print("IPK: ")
    fmt.Scan(&dataMhs[i].ipk)
}

// Mendapatkan indeks IPK tertinggi
idxTertinggi := indeksIPKTertinggi(dataMhs, n)

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

```

Scrennshot output

A screenshot of a terminal window with a dark background. The terminal has tabs at the top: 'PROBLEMS (8)', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is active), and 'SEARCH ERROR'. The output text is as follows:

```
NIM: 211102279
Kelas: 1
Jurusan: SistemInformasi
IPK: 3.88

Mahasiswa dengan IPK tertinggi:
Nama: Martin
NIM: 2311102269
Kelas: 6
Jurusan: Informatika
IPK: 4.00
PS C:\Users\MARTIN SIMBOLON>
```

Deskripsi output

Program di atas dirancang untuk mengelola data mahasiswa dan mencari mahasiswa dengan IPK tertinggi. Program ini menggunakan tipe data `struct` untuk menyimpan informasi tentang mahasiswa, yang terdiri dari nama, NIM, kelas, jurusan, dan IPK. Program meminta input dari pengguna untuk menentukan jumlah mahasiswa (maksimal 2023) dan kemudian mengisi data mahasiswa tersebut, termasuk nama, NIM, kelas, jurusan, dan IPK. Setelah semua data dimasukkan, program mencari indeks mahasiswa dengan IPK tertinggi menggunakan fungsi `indeksIPKTertinggi`, yang membandingkan nilai IPK setiap mahasiswa. Setelah menemukan mahasiswa dengan IPK tertinggi, program menampilkan informasi lengkap mahasiswa tersebut, termasuk nama, NIM, kelas, jurusan, dan nilai IPK tertinggi. Program ini memanfaatkan array untuk menyimpan data mahasiswa dan dapat digunakan untuk aplikasi manajemen data akademik sederhana.

UNGUIDED

1. Sourcecode

```
package main

import (
    "fmt"
)

func main() {

    var N int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&N)

    if N > 1000 {
        fmt.Println("Jumlah anak kelinci tidak boleh lebih dari 1000!")
        return
    }

    berat := make([]float64, N)

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

    minBerat := berat[0]
    maxBerat := berat[0]

    for i := 1; i < N; i++ {
        if berat[i] > maxBerat {
            maxBerat = berat[i]
        }
        if berat[i] < minBerat {
            minBerat = berat[i]
        }
    }
}
```

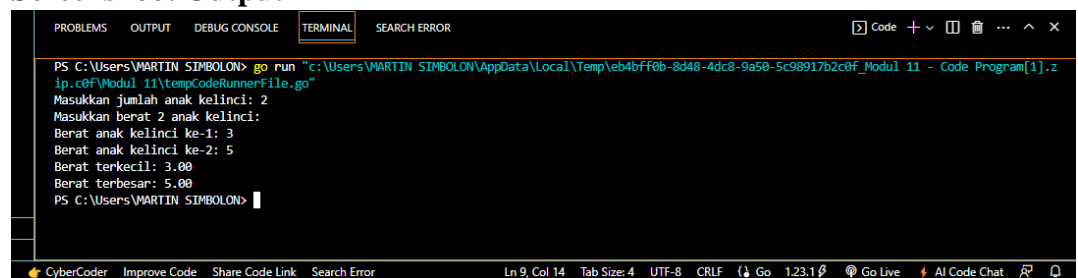
```

    }

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

```

Screenshoot Output



```

PS C:\Users\MARTIN SIMBOLON> go run "c:\Users\MARTIN SIMBOLON\AppData\Local\Temp\eb4bff0b-8d48-4dc8-9a50-5c98917b2c0f_Modul 11 - Code Program[1].2
ip.c0fVModul 11\tempCodeRunnerFile.go"
Masukkan jumlah anak kelinci: 2
Masukkan berat 2 anak kelinci:
Berat anak kelinci ke-1: 3
Berat anak kelinci ke-2: 5
Berat terkecil: 3.00
Berat terbesar: 5.00
PS C:\Users\MARTIN SIMBOLON>

```

Deskripsi Program

Program di atas adalah sebuah aplikasi sederhana menggunakan bahasa Go yang bertujuan untuk mencari berat terkecil dan terbesar dari sekumpulan data berat anak kelinci. Program meminta pengguna untuk memasukkan jumlah anak kelinci (NNN) dan berat masing-masing kelinci yang akan disimpan dalam array berukuran NNN. Dengan menggunakan algoritma iterasi, program membandingkan setiap berat dalam array untuk menentukan nilai maksimum dan minimum. Jika jumlah anak kelinci yang dimasukkan lebih dari 1000, program akan memberikan peringatan dan menghentikan eksekusi. Hasil akhir berupa dua nilai, yaitu berat terkecil dan terbesar, ditampilkan dalam format desimal dua angka. Program ini dirancang untuk memproses data dengan efisiensi $O(n)O(n)O(n)$ dan memberikan keluaran yang cepat dan akurat.

2. Sourcecode

```
package main

import (
    "fmt"
)

func main() {

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

    if x > 1000 || y > 1000 {
        fmt.Println("Jumlah ikan atau wadah tidak boleh lebih dari 1000!")
        return
    }

    ikan := make([]float64, x)
    fmt.Printf("Masukkan berat %d ikan:\n", x)
    for i := 0; i < x; i++ {
        fmt.Printf("Berat ikan ke-%d: ", i+1)
        fmt.Scan(&ikan[i])
    }

    totalPerWadah := make([]float64, y)
    for i := 0; i < x; i++ {

        indexWadah := i % y
        totalPerWadah[indexWadah] += ikan[i]
    }

    rataRataPerWadah := make([]float64, y)
    for i := 0; i < y; i++ {

        jumlahIkanDiWadah := (x + i) / y
        if jumlahIkanDiWadah > 0 {
            rataRataPerWadah[i] = totalPerWadah[i] /
float64(jumlahIkanDiWadah)
        } else {
            rataRataPerWadah[i] = 0
        }
    }
}
```

```

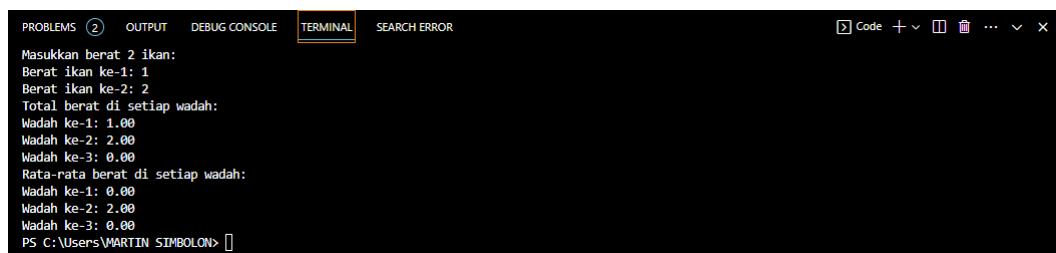
    }
}

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

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

```

Screenshoot output



```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL SEARCH ERROR
Masukkan berat 2 ikan:
Berat ikan ke-1: 1
Berat ikan ke-2: 2
Total berat di setiap wadah:
Wadah ke-1: 1.00
Wadah ke-2: 2.00
Wadah ke-3: 0.00
Rata-rata berat di setiap wadah:
Wadah ke-1: 0.00
Wadah ke-2: 2.00
Wadah ke-3: 0.00
PS C:\Users\WARTIN SIMBOLON>

```

Deskripsi program

Program di atas dirancang untuk menghitung total berat ikan dan rata-rata berat ikan di setiap wadah berdasarkan jumlah ikan (xxx) dan jumlah wadah (yyy). Program meminta pengguna untuk memasukkan data jumlah ikan dan wadah, diikuti dengan berat masing-masing ikan, yang kemudian didistribusikan secara berurutan ke wadah menggunakan logika modulus ($i\%y$ \ $y\%y$). Total berat untuk setiap wadah dihitung dengan menjumlahkan berat ikan yang masuk ke wadah tersebut, sedangkan rata-rata berat dihitung dengan membagi total berat wadah dengan jumlah ikan yang didistribusikan ke dalamnya. Hasil akhirnya berupa daftar total berat dan rata-rata berat untuk masing-masing wadah, yang dicetak dengan format desimal dua angka. Program ini dirancang untuk memastikan distribusi ikan ke wadah berjalan merata dan efisien, dengan kapasitas hingga 1000 ikan dan 1000 wadah.

3. Sourcecode

```
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 {
    total := 0.0
    for i := 0; i < n; i++ {
        total += arrBerat[i]
    }
    return total / float64(n)
}

func main() {
    var n int
    var berat 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(&berat[i])
    }
}
```



```

hitungMinMax(berat, n, &bMin, &bMax)
rataRata := rerata(berat, n)

fmt.Printf("Berat balita terkecil: %.2f kg\n", bMin)
fmt.Printf("Berat balita terbesar: %.2f kg\n", bMax)
fmt.Printf("Rata-rata berat balita: %.2f kg\n", rataRata)
}

```

Scrennshoot output

```

PS C:\Users\MARTIN SIMBOLON> go run "c:\Users\MARTIN SIMBOLON\AppData\Local\Temp\eb4bff0b-8d48-4dc8-9a50-5c94
ip.c0f\Modul 11\TipeDasar.go"
Masukkan banyak data berat balita: 4
Masukkan berat balita ke-1: 4.5
Masukkan berat balita ke-2: 5.4
Masukkan berat balita ke-3: 6.1
Masukkan berat balita ke-4: 3.5
Berat balita terkecil: 3.50 kg
Berat balita terbesar: 6.10 kg
Rata-rata berat balita: 4.88 kg
PS C:\Users\MARTIN SIMBOLON>

```

Deskripsi Program

Program ini dirancang untuk membantu petugas Pos Pelayanan Terpadu (Posyandu) mencatat dan menganalisis data berat balita dalam satuan kilogram. Program memungkinkan pengguna memasukkan jumlah balita dan berat masing-masing balita, yang kemudian disimpan dalam array statis dengan kapasitas maksimum 100 elemen. Program menggunakan dua fungsi utama: `hitungMinMax` untuk menghitung berat balita terkecil dan terbesar dalam array, serta `rerata` untuk menghitung rata-rata berat balita. Dengan pendekatan iterasi, program memproses data untuk menampilkan hasil berupa nilai minimum, maksimum, dan rata-rata berat balita, sehingga memudahkan petugas dalam melakukan evaluasi data kesehatan secara efisien dan akurat.