

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL 11

PENCARIAN NILAI EXTRIM PADA HIMPUNAN DATA



Disusun Oleh :

HANIF REYHAN ZHAFRAN ADRYTONA / 2311102266

11 IF 06

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pencarian nilai ekstrem, yaitu nilai maksimum dan minimum, merupakan proses untuk menentukan elemen terbesar dan terkecil dalam sebuah himpunan data. Operasi ini sering digunakan dalam berbagai bidang, seperti analisis data, statistik, dan pemrosesan real-time. Dalam pemrograman, pencarian nilai ekstrem dilakukan dengan cara membandingkan setiap elemen dalam struktur data tertentu, seperti array atau slice. Bahasa pemrograman Go, yang terkenal karena kesederhanaannya, menyediakan fitur dasar yang sangat cocok untuk implementasi pencarian nilai ekstrem ini.

Algoritma pencarian nilai ekstrem dimulai dengan menetapkan elemen pertama sebagai nilai minimum dan maksimum sementara. Kemudian, algoritma melakukan iterasi melalui setiap elemen dalam data, membandingkan nilai elemen tersebut dengan nilai minimum dan maksimum yang telah ditetapkan. Jika ditemukan elemen yang lebih kecil, nilai minimum diperbarui; sebaliknya, jika ada elemen yang lebih besar, nilai maksimum juga diperbarui. Proses ini terus berlangsung hingga semua elemen selesai diperiksa. Dengan kompleksitas waktu $O(n)$, algoritma ini cukup efisien untuk himpunan data kecil hingga sedang.

Implementasi pencarian nilai ekstrem dalam Go dapat dilakukan menggunakan perulangan sederhana. Misalnya, sebuah slice berisi angka-angka dapat diproses menggunakan iterasi `for` dan operator logika untuk menentukan nilai ekstrem. Hasil akhirnya berupa nilai minimum dan maksimum yang ditemukan selama iterasi. Pendekatan ini tidak hanya mudah dipahami, tetapi juga fleksibel untuk diterapkan dalam berbagai aplikasi, seperti monitoring data, deteksi anomali, atau pemrosesan dataset secara real-time. Keandalan dan efisiensi algoritma ini menjadikannya salah satu operasi dasar yang sering digunakan dalam pengolahan data.

II. GUIDED

Guided 1.0

Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona
package main

import "fmt"

type arrInt [2023]int

func terkecil(tabInt arrInt, n int) int {
    var idx int = 0
    for j := 1; j < n; j++ {
        if tabInt[idx] > tabInt[j] {
            idx = j
        }
    }
    return idx
}

func main() {
    var n int
    var tab arrInt

    fmt.Print("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n)

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

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

    idxMin := terkecil(tab, n)

    fmt.Println("Nilai terkecil dalam array adalah:",
tab[idxMin], "pada indeks:", idxMin)
```

```
}
}
```

Screenshoot Output



```
PROBLEMS OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE
PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 11\GUIDED.1.go"
Masukkan jumlah elemen (maks 2023): 5
Masukkan elemen-elemen array:
Elemen ke-1: 5
Elemen ke-1: 5
Elemen ke-2: 4
Elemen ke-2: 4
Elemen ke-3: 3
Elemen ke-4: 2
Elemen ke-5: 1
Nilai terkecil dalam array adalah: 1 pada indeks: 4
PS C:\Users\M S I>
```

Deskripsi Program

Program di atas bertujuan untuk menemukan nilai terkecil dalam sebuah array beserta indeksinya. Pengguna diminta untuk memasukkan jumlah elemen array (dengan batas maksimum 2023) dan elemen-elemen array itu sendiri. Setelah data diinput, program memanggil fungsi `terkecil`, yang melakukan pencarian nilai terkecil dengan membandingkan setiap elemen dalam array. Fungsi ini memanfaatkan algoritma iterasi sederhana, di mana elemen pertama diinisialisasi sebagai nilai terkecil sementara, lalu dibandingkan dengan elemen lainnya. Jika ditemukan elemen yang lebih kecil, indeks nilai terkecil diperbarui. Setelah iterasi selesai, fungsi mengembalikan indeks dari elemen terkecil.

Hasil dari program ini adalah nilai terkecil dalam array dan indeksinya, yang kemudian ditampilkan kepada pengguna. Program ini mencakup validasi jumlah elemen array untuk memastikan input sesuai dengan batas yang ditentukan. Dengan implementasi yang sederhana, program ini cocok digunakan untuk latihan dasar manipulasi array dalam bahasa Go dan memahami konsep iterasi, perbandingan, serta pengelolaan input dan output.

Guided 2.0

Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

package main

import "fmt"

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

type arrMhs [2023]mahasiswa

func ipkTertinggi(T arrMhs, n int) float64 {
    var tertinggi float64 = T[0].ipk
    for j := 1; j < n; j++ {
        if tertinggi < T[j].ipk {
            tertinggi = T[j].ipk
        }
    }
    return tertinggi
}

func main() {
    var n int
    var dataMhs arrMhs

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

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

    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: ")
    }
}
```

```

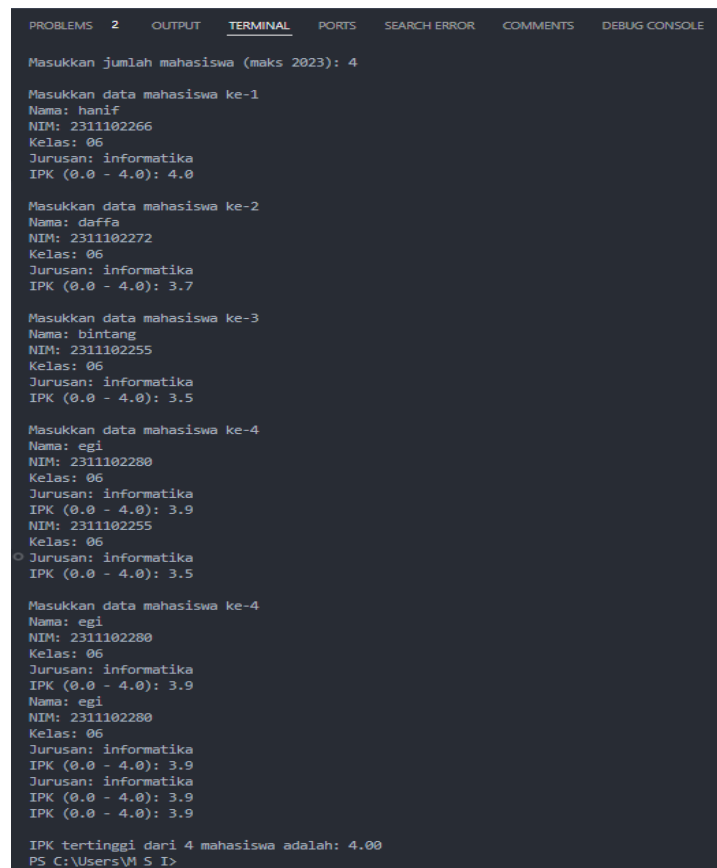
        fmt.Scan(&dataMhs[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scan(&dataMhs[i].jurusan)
        fmt.Print("IPK (0.0 - 4.0): ")
        fmt.Scan(&dataMhs[i].ipk)

        if dataMhs[i].ipk < 0.0 || dataMhs[i].ipk > 4.0 {
            fmt.Println("IPK tidak valid. Harus antara 0.0 dan
4.0.")
            return
        }
    }

    tertinggi := ipkTertinggi(dataMhs, n)
    fmt.Printf("\nIPK tertinggi dari %d mahasiswa adalah: %.2f\n",
n, tertinggi)
}

```

Screenshoot Output



```

PROBLEMS 2 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE

Masukkan jumlah mahasiswa (maks 2023): 4

Masukkan data mahasiswa ke-1
Nama: hanif
NIM: 2311102266
Kelas: 06
Jurusan: informatika
IPK (0.0 - 4.0): 4.0

Masukkan data mahasiswa ke-2
Nama: daffa
NIM: 2311102272
Kelas: 06
Jurusan: informatika
IPK (0.0 - 4.0): 3.7

Masukkan data mahasiswa ke-3
Nama: bintang
NIM: 2311102255
Kelas: 06
Jurusan: informatika
IPK (0.0 - 4.0): 3.5

Masukkan data mahasiswa ke-4
Nama: egi
NIM: 2311102280
Kelas: 06
Jurusan: informatika
IPK (0.0 - 4.0): 3.9
NIM: 2311102255
Kelas: 06
Jurusan: informatika
IPK (0.0 - 4.0): 3.5

Masukkan data mahasiswa ke-4
Nama: egi
NIM: 2311102280
Kelas: 06
Jurusan: informatika
IPK (0.0 - 4.0): 3.9
Nama: egi
NIM: 2311102280
Kelas: 06
Jurusan: informatika
IPK (0.0 - 4.0): 3.9
Jurusan: informatika
IPK (0.0 - 4.0): 3.9
IPK (0.0 - 4.0): 3.9

IPK tertinggi dari 4 mahasiswa adalah: 4.00
PS C:\Users\VM S I>

```

Deskripsi Program

Program di atas adalah sebuah program dengan bahasa go, program ini bertujuan untuk mencari **IPK tertinggi** dari sejumlah data mahasiswa yang dimasukkan pengguna. Data mahasiswa disimpan dalam bentuk struct yang terdiri dari atribut seperti nama, NIM, kelas, jurusan, dan IPK. Untuk menyimpan data banyak mahasiswa, digunakan array `arrMhs` dengan kapasitas maksimum 2023 elemen. Pengguna diminta memasukkan jumlah mahasiswa terlebih dahulu, dengan batas validasi antara 1 hingga 2023. Setelah itu, data setiap mahasiswa dimasukkan satu per satu, termasuk nama, NIM, kelas, jurusan, dan IPK. Program juga dilengkapi validasi untuk memastikan nilai IPK yang dimasukkan berada dalam rentang 0.0 hingga 4.0. Jika input tidak valid, program akan memberikan pesan kesalahan dan menghentikan eksekusi.

Proses pencarian IPK tertinggi dilakukan melalui fungsi `ipkTertinggi`. Fungsi ini bekerja dengan membandingkan nilai IPK setiap mahasiswa dalam array menggunakan perulangan. IPK pertama diinisialisasi sebagai nilai tertinggi awal, lalu dibandingkan dengan IPK mahasiswa lainnya. Jika ditemukan IPK yang lebih besar, nilai tertinggi diperbarui. Setelah iterasi selesai, program mengembalikan nilai IPK tertinggi yang ditemukan. Hasil ini kemudian ditampilkan kepada pengguna, lengkap dengan jumlah mahasiswa yang dianalisis. Dengan struktur yang sederhana dan efisien, program ini memberikan gambaran tentang cara memproses data terstruktur dan mencari nilai maksimum dalam bahasa Go, serta memastikan data yang diolah valid dan relevan.

III. UNGUIDED

Soal Studi Case 1

Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak kelinci yang akan dijual. Masukan terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat N yang menyatakan banyaknya anak kelinci yang akan ditimbang beratnya. Selanjutnya N bilangan riil berikutnya adalah berat dari anak kelinci yang akan dijual. Keluaran terdiri dari dua buah bilangan riil yang menyatakan berat kelinci terkecil dan terbesar

Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

package main

import (
    "fmt"
    "math"
)

func main() {
    var N int
    var berat float64

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

    if N <= 0 || N > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")
        return
    }
    beratKelinci := make([]float64, N)
    fmt.Println("berat anak kelinci:")
    for i := 0; i < N; i++ {
        fmt.Printf("%d: ", i+1)
        fmt.Scan(&berat)
        beratKelinci[i] = berat
    }
}
```



```

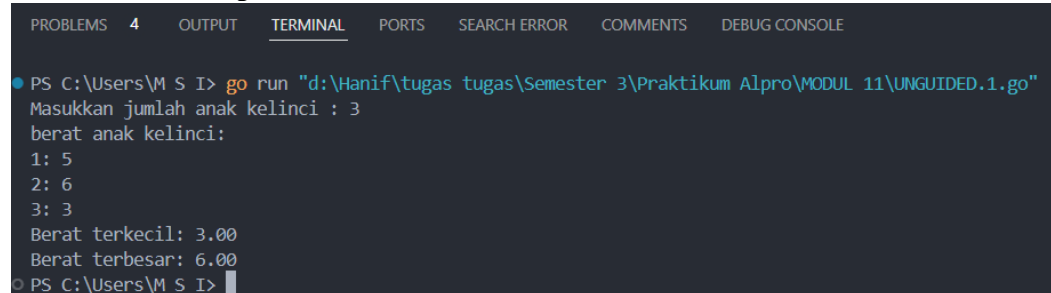
    terkecil := math.MaxFloat64
    terbesar := -math.MaxFloat64
    for _, b := range beratKelinci {
        if b < terkecil {
            terkecil = b
        }
        if b > terbesar {
            terbesar = b
        }
    }
    fmt.Printf("Berat terkecil: %.2f\n", terkecil)
    fmt.Printf("Berat terbesar: %.2f\n", terbesar)
}

    stdDeviasi := math.Sqrt(varianceSum /
float64(len(array)))
    fmt.Printf("Standar deviasi elemen array: %.2f\n",
stdDeviasi)

    fmt.Print("\nMasukkan bilangan untuk mengetahui
frekuensinya: ")
    fmt.Scan(&target)
    count := 0
    for _, v := range array {
        if v == target {
            count++
        }
    }
    fmt.Printf("Frekuensi bilangan %d dalam array: %d\n",
target, count)
}

```

Screenshoot Output



```

PROBLEMS 4 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE
PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 11\UNGUIDED.1.go"
Masukkan jumlah anak kelinci : 3
berat anak kelinci:
1: 5
2: 6
3: 3
Berat terkecil: 3.00
Berat terbesar: 6.00
PS C:\Users\M S I>

```

Deskripsi Program

Program di atas adalah sebuah program dengan bahasa go, pada program ini dirancang untuk menghitung berat terkecil dan terbesar dari sejumlah anak kelinci yang dimasukkan oleh pengguna. Pengguna diminta untuk memasukkan jumlah anak kelinci terlebih dahulu, yang kemudian divalidasi untuk memastikan berada dalam rentang 1 hingga 1000. Setelah jumlah anak kelinci valid, program akan meminta pengguna untuk memasukkan berat setiap anak kelinci satu per satu. Berat-berat ini disimpan dalam slice `beratKelinci`, yang memungkinkan fleksibilitas untuk menampung data dengan jumlah elemen yang dinamis sesuai input pengguna.

Setelah semua berat dimasukkan, program melanjutkan untuk mencari berat terkecil dan terbesar dengan cara membandingkan setiap nilai berat dalam slice. Untuk mencari nilai terkecil, program menginisialisasi variabel `terkecil` dengan nilai maksimum yang mungkin (`math.MaxFloat64`), sementara untuk mencari nilai terbesar, variabel `terbesar` diinisialisasi dengan nilai minimum (`-math.MaxFloat64`). Program kemudian melakukan iterasi terhadap setiap elemen dalam slice dan memperbarui nilai `terkecil` dan `terbesar` jika ditemukan nilai yang lebih kecil atau lebih besar.

Setelah proses pencarian selesai, program menampilkan hasil berat terkecil dan terbesar dengan format dua angka desimal untuk memudahkan pembacaan. Program ini menggunakan teknik pencarian yang efisien dengan kompleksitas $O(n)$, sehingga dapat menangani hingga 1000 data tanpa kendala. Dengan struktur yang sederhana dan penggunaan validasi input, program ini cocok untuk aplikasi yang membutuhkan pemrosesan data numerik dalam jumlah sedang.

Unguided 2

Soal Studi Case 2

Sebuah program digunakan untuk menentukan tarif ikan yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat ikan yang akan dijual. Masukan terdiri dari dua baris, yang mana baris pertama terdiri dari dua bilangan bulat x dan y . Bilangan x menyatakan banyaknya ikan yang akan dijual, sedangkan y adalah banyaknya ikan yang akan dimasukkan ke dalam wadah. Baris kedua terdiri dari sejumlah x bilangan riil yang menyatakan banyaknya ikan yang akan dijual. Keluaran terdiri dari dua baris. Baris pertama adalah kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah (jumlah wadah tergantung pada nilai x dan y , urutan ikan yang dimasukkan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

package main

import "fmt"

func main() {

    fmt.Println("masukan jumlah ikan dan kapasitas wadah pada (x)&(y) :")

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

    fishWeights := make([]float64, x)
    fmt.Printf("Masukkan %d berat ikan (pisahkan dengan spasi): ",
x)
    for i := 0; i < x; i++ {
        fmt.Scan(&fishWeights[i])
    }

    buckets := make([][]float64, y)
    for i, weight := range fishWeights {
        bucketIndex := i % y
        buckets[bucketIndex] = append(buckets[bucketIndex], weight)
    }

    totalWeights := make([]float64, y)
    averageWeights := make([]float64, y)
    for i := 0; i < y; i++ {
        var totalWeight float64
        for _, weight := range buckets[i] {
            totalWeight += weight
        }
        totalWeights[i] = totalWeight
        if len(buckets[i]) > 0 {
            averageWeights[i] = totalWeight /
float64(len(buckets[i]))
        }
    }

    fmt.Println("\nHasil Distribusi:")
}
```

```

    fmt.Println("Total berat di setiap wadah:")
    for i, total := range totalWeights {
        fmt.Printf("Wadah %d: %.2f\n", i+1, total)
    }
    fmt.Println("\nRata-rata berat ikan di setiap wadah:")
    for i, avg := range averageWeights {
        fmt.Printf("Wadah %d: %.2f\n", i+1, avg)
    }
}

```

Screenshoot Output

```

PROBLEMS 6 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE
● PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 11\UNGUIDED.2.go"
masukan jumlah ikan dan kapasitas wadah pada (x)&(y) :
Masukkan jumlah ikan (x) dan jumlah wadah (y): 5 10
Masukkan 5 berat ikan (pisahkan dengan spasi): 1 2 3 4 5

Hasil Distribusi:
Total berat di setiap wadah:
Wadah 1: 1.00
Wadah 2: 2.00
Wadah 3: 3.00
Wadah 4: 4.00
Wadah 5: 5.00
Wadah 6: 0.00
Wadah 7: 0.00
Wadah 8: 0.00
Wadah 9: 0.00
Wadah 10: 0.00

Rata-rata berat ikan di setiap wadah:
Wadah 1: 1.00
Wadah 2: 2.00
Wadah 3: 3.00
Wadah 4: 4.00
Wadah 5: 5.00
Wadah 6: 0.00
Wadah 7: 0.00
Wadah 8: 0.00
Wadah 9: 0.00
Wadah 10: 0.00
○ PS C:\Users\M S I>

```

Deskripsi Program

Program di atas adalah sebuah program dengan bahasa go yang digunakan untuk mendistribusikan berat ikan ke dalam beberapa wadah dan menghitung total serta rata-rata berat ikan di setiap wadah. Pertama, pengguna diminta untuk memasukkan jumlah ikan (x) dan jumlah wadah (y). Setelah itu, program meminta pengguna untuk memasukkan berat ikan satu per satu, yang kemudian disimpan dalam sebuah slice `fishWeights`. Program kemudian menggunakan slice ini untuk mendistribusikan berat ikan ke dalam wadah dengan cara **round-robin**

(bergilir), di mana ikan pertama dimasukkan ke wadah pertama, ikan kedua ke wadah kedua, dan seterusnya hingga semua ikan terdistribusi.

Setelah proses distribusi selesai, program melanjutkan untuk menghitung total berat ikan di setiap wadah. Program menjumlahkan semua berat ikan dalam setiap wadah menggunakan perulangan dan menyimpannya dalam slice `totalWeights`. Selanjutnya, program menghitung rata-rata berat ikan di setiap wadah dengan membagi total berat ikan di wadah tersebut dengan jumlah ikan yang ada di wadah tersebut, dan hasilnya disimpan dalam slice `averageWeights`. Ini memungkinkan program untuk memberikan gambaran tidak hanya tentang total berat, tetapi juga distribusi berat yang lebih merata di antara wadah-wadah yang ada.

Pada akhir program, hasil distribusi ditampilkan dengan menampilkan total berat dan rata-rata berat ikan di setiap wadah. Program memastikan bahwa setiap wadah menerima ikan secara adil meskipun jumlah ikan dan wadah tidak selalu dapat dibagi rata. Dengan menggunakan struktur data slice dan operasi modulus untuk distribusi ikan, program ini bekerja dengan efisien dan fleksibel, memberikan output yang informatif kepada pengguna mengenai distribusi dan rata-rata berat ikan dalam wadah.

UNGUIDED 3

Soal Studi Case 3

Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukkan data tersebut ke dalam array. Dari data yang diperoleh akan dicari berat balita terkecil, terbesar, dan reratanya. Buatlah program dengan spesifikasi subprogram sebagai berikut:

Sourcecode

```
// 2311102266_Hanif Reyhan Zhafran Arytona

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 arrBerat arrBalita
    var bMin, bMax float64

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

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

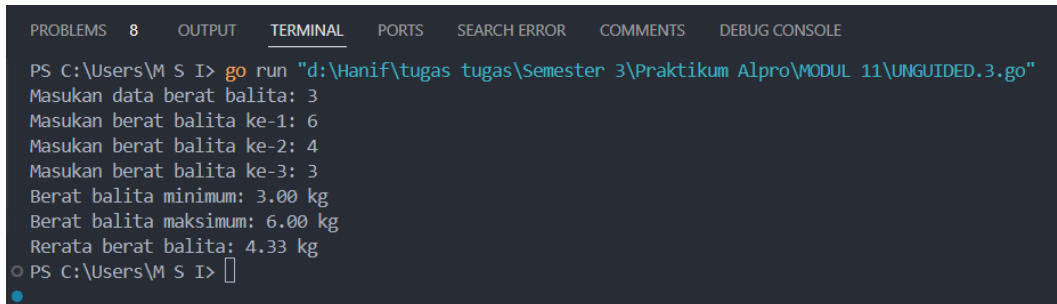
    hitungMinMax(arrBerat, n, &bMin, &bMax)

    rata := 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", rata)
}

```

Screenshoot Output



```
PROBLEMS 8 OUTPUT TERMINAL PORTS SEARCH ERROR COMMENTS DEBUG CONSOLE
PS C:\Users\M S I> go run "d:\Hanif\tugas tugas\Semester 3\Praktikum Alpro\MODUL 11\UNGUIDED.3.go"
Masukan data berat balita: 3
Masukan berat balita ke-1: 6
Masukan berat balita ke-2: 4
Masukan berat balita ke-3: 3
Berat balita minimum: 3.00 kg
Berat balita maksimum: 6.00 kg
Rerata berat balita: 4.33 kg
PS C:\Users\M S I> 
```

Deskripsi Program

Program di atas berfungsi untuk menghitung berat balita minimum dengan implementasi dengan bahasa go, maksimum, dan rata-rata berdasarkan input yang diberikan oleh pengguna. Pertama, pengguna diminta untuk memasukkan jumlah balita dan berat masing-masing balita. Program kemudian menyimpan berat-berat tersebut dalam array `arrBerat`. Setelah itu, fungsi `hitungMinMax` digunakan untuk mencari berat balita terkecil (minimum) dan terbesar (maksimum) dengan memeriksa setiap elemen dalam array. Fungsi `rerata` menghitung rata-rata berat balita dengan menjumlahkan semua elemen dalam array dan membaginya dengan jumlah balita.

Setelah semua perhitungan selesai, program menampilkan hasil berupa berat balita minimum, maksimum, dan rata-rata dengan format dua angka desimal. Program ini menggunakan array untuk menyimpan data dan pointer untuk memperbarui nilai-nilai minimum dan maksimum secara efisien. Dengan pendekatan yang sederhana dan langsung, program ini berguna untuk aplikasi pemantauan berat balita, seperti dalam pengolahan data kesehatan anak, dan dapat menangani hingga 100 balita dengan akurat.