

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL XI

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA



Disusun Oleh :

Muhammad Ihab Aufa Rafi / 2311102226

S1IF-11-06

Dosen Pengampu :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

1.1 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 ini 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 ekstrim tersebut dengan data yang dicek.
- 3) Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid.

Berikut ini adalah notasi dalam pseudocode dan bahasa Go, misalnya untuk pencarian nilai terbesar atau maksimum:

	Notasi Algoritma	Notasi dalam bahasa Go
1	$\text{max} \leftarrow 1$	$\text{max} = 0$
2	$i \leftarrow 2$	$i = 1$
3	while $i \leq n$ do	for $i < n$ {
4	if $a[i] > a[\text{max}]$ then	if $a[i] > a[\text{max}]$ {
5	$\text{max} \leftarrow i$	$\text{max} = i$
6	endif	}
7	$i \leftarrow i + 1$	$i = i + 1$
8	endwhile	}

1.2 Pencarian Nilai Ekstrim pada Array Bertipe Data Dasar

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut. Perhatikan potongan program dalam bahasa Go berikut ini!

```
5  type arrInt [2023]int
..  ...
15
16  func terkecil_1(tabInt arrInt, n int) int {
17  /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
18  bilangan bulat */
19      var min int = tabInt[0]           // min berisi data pertama
20      var j int = 1                    // pencarian dimulai dari data berikutnya
21      for j < n {
22          if min > tabInt[j] {          // pengecekan apakah nilai minimum valid
23              min = tabInt[j]          // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return min                       // returnkan nilai minimumnya
28  }
```

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada bahasa Go di mulai dari nol atau "0". Pada algoritma pencarian, yang terpenting adalah posisi atau indeks dari nilai yang dicari dalam kumpulan data atau array. Oleh karena itu modifikasi pada program di atas dapat dilihat pada potongan program berikut ini!

```
..  ...
5  type arrInt [2023]int
..  ...
15
16  func terkecil_2(tabInt arrInt, n int) int {
17  /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi n bilangan bulat*/
18      var idx int = 0                  // idx berisi indeks data pertama
19      var j int = 1                    // pencarian dimulai dari data berikutnya
20      for j < n {
21          if tabInt[idx] > tabInt[j] {  // pengecekan apakah nilai minimum valid
22              idx = j                  // update nilai minimum dengan yang valid
23          }
24          j = j + 1
25      }
26      return idx                       // returnkan indeks nilai minimumnya
27  }
```

1.3 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.

```
.. ...
5  type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
.. ...
15
16 func IPK_1(T arrMhs, n int) float64 {
17     /* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
18     n mahasiswa */
19     var tertinggi float64 = T[0].ipk
20     var j int = 1
21     for j < n {
22         if tertinggi < T[j].ipk {
23             tertinggi = T[j].ipk
24         }
25         j = j + 1
26     }
27     return tertinggi
28 }
```

Apabila diperhatikan potongan program di atas, maka kita akan memperoleh `ipk tertinggi`, tetapi kita tidak memperoleh identitas mahasiswa dengan `ipk tertinggi` tersebut. Maka seperti penjelasan yang sudah diberikan sebelumnya, maka pencarian yang dilakukan bisa mengembalikan indeks mahasiswa dengan `ipk tertinggi` tersebut. Berikut ini adalah modifikasinya!

```
.. ...
5  type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
.. ...
15
16 func IPK_2(T arrMhs, n int) int {
17     /* mengembalikan indeks mahasiswa yang memiliki ipk tertinggi pada array T yang
18     berisi n mahasiswa */
19     var idx int = 0
20     var j int = 1
21     for j < n {
22         if T[idx].ipk < T[j].ipk {
23             idx = j
24         }
25         j = j + 1
26     }
27     return idx
28 }
```

Sehingga melalui algoritma di atas, identitas mahasiswa dapat diperoleh, misalnya `T[idx].nama`, `T[idx].nim`, `T[idx].kelas`, hingga `T[idx].jurusan`.

II. GUIDED

1. Soal Studi Case

Contoh program nilai minimum.

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
```

```

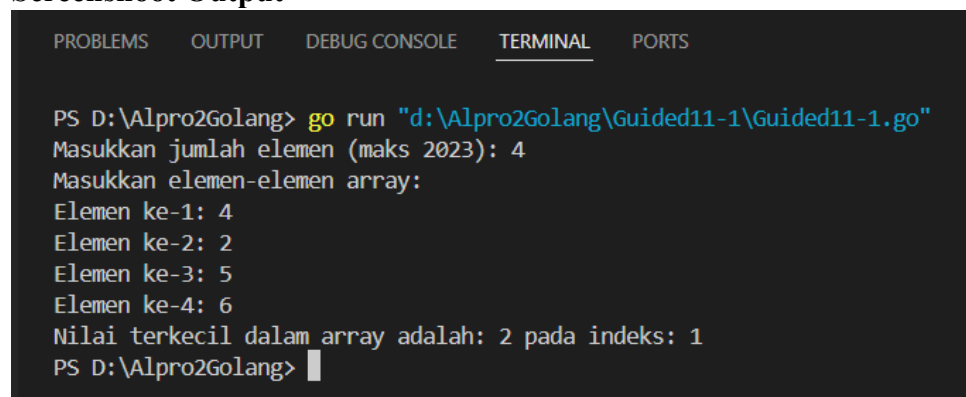
    fmt.Println("Masukkan elemen-elemen array:")
    for i := 0; i < n; i++ {
        fmt.Print("Elemen ke-", i+1, ": ")
        fmt.Scan(&tab[i])
    }

    // 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



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Guided11-1\Guided11-1.go"
Masukkan jumlah elemen (maks 2023): 4
Masukkan elemen-elemen array:
Elemen ke-1: 4
Elemen ke-2: 2
Elemen ke-3: 5
Elemen ke-4: 6
Nilai terkecil dalam array adalah: 2 pada indeks: 1
PS D:\Alpro2Golang>

```

Deskripsi Program

Program ini merupakan implementasi untuk mencari elemen terkecil dalam sebuah array dengan panjang maksimum 2023. Program menggunakan tipe data khusus `arrInt` untuk mendeklarasikan array dengan panjang tetap sebesar 2023 elemen. Fungsi `terkecil` berperan dalam menentukan indeks elemen terkecil di dalam array. Fungsi ini bekerja dengan membandingkan elemen array secara berulang, dimulai dari indeks pertama hingga elemen terakhir. Jika ditemukan elemen yang lebih kecil, indeks elemen tersebut disimpan dan digunakan sebagai hasil.

Di dalam fungsi `main`, pengguna diminta untuk memasukkan jumlah elemen array terlebih dahulu, dengan batas validasi antara 1 hingga 2023. Setelah itu, pengguna akan mengisi nilai-nilai elemen array satu per satu. Setelah data dimasukkan, program memanggil fungsi `terkecil` untuk menemukan indeks elemen terkecil.

2. Soal Studi Case

Contoh program nilai maksimum.

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 IPK tertinggi dalam array mahasiswa
func ipk(T arrMhs, n int) float64 {
    var tertinggi float64 = T[0].ipk
    var j int = 1
    for j < n {
        if tertinggi < T[j].ipk {
            tertinggi = T[j].ipk
        }
        j = j + 1
    }
    return tertinggi
}

// Fungsi main untuk mengisi data mahasiswa dan mencari IPK
tertinggi
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)
}

// Mencari dan menampilkan IPK tertinggi
tertinggi := ipk(dataMhs, n)
fmt.Printf("\nIPK tertinggi dari %d mahasiswa adalah:
%.2f\n", n, tertinggi)
}

```

Screenshoot Output


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Guided11-2\Guided11-2.go"
Masukkan jumlah mahasiswa (maks 2023): 3

Masukkan data mahasiswa ke-1
Nama: ihab
NIM: 2311102226
Kelas: 11-06
Jurusan: informatika
IPK: 3.5

Masukkan data mahasiswa ke-2
Nama: aufa
NIM: 231110xxxx
Kelas: 11-03
Jurusan: informatika
IPK: 4.0

Masukkan data mahasiswa ke-3
Nama: rafi
NIM: 2311102xxx
Kelas: 11-04
Jurusan: informatika
IPK: 3.2

IPK tertinggi dari 3 mahasiswa adalah: 4.00
PS D:\Alpro2Golang> 
```

Deskripsi Program

Program ini merupakan implementasi untuk mencari IPK (Indeks Prestasi Kumulatif) tertinggi di antara data mahasiswa yang dimasukkan. Program menggunakan struct mahasiswa untuk merepresentasikan data setiap mahasiswa dengan atribut **nama**, **nim**, **kelas**, **jurusan**, dan **ipk**. Selain itu, tipe data khusus **arrMhs** digunakan untuk mendefinisikan array dengan kapasitas maksimum 2023 mahasiswa.

Fungsi **ipk** bertugas menghitung IPK tertinggi dari data mahasiswa yang ada. Fungsi ini melakukan iterasi melalui elemen-elemen array mahasiswa, membandingkan IPK setiap mahasiswa dengan nilai tertinggi yang sudah ditemukan. Jika ditemukan IPK yang lebih besar, nilai tersebut disimpan sebagai yang tertinggi hingga iterasi selesai.

Di dalam fungsi **main**, pengguna diminta untuk memasukkan jumlah mahasiswa terlebih dahulu. Jumlah tersebut harus berada dalam rentang 1 hingga 2023. Jika jumlah yang dimasukkan tidak valid, program akan menampilkan pesan kesalahan dan berhenti. Setelah jumlah mahasiswa

valid, program meminta pengguna untuk mengisi data setiap mahasiswa, termasuk nama, NIM, kelas, jurusan, dan IPK.

III. UNGUIDED

1. Soal Studi Case

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

```
package main

import "fmt"

func masukanBeratKelinci(n int) []float64 {
    beratKelinci := make([]float64, n)
    for i := 0; i < n; i++ {
        fmt.Printf("Berat Kelinci %d: ", i+1)
        fmt.Scan(&beratKelinci[i])
    }
    return beratKelinci
}

func cariBeratTerkecilTerbesar(beratKelinci []float64) (float64, float64) {
    beratMin := beratKelinci[0]
    beratMax := beratKelinci[0]

    for i := 1; i < len(beratKelinci); i++ {
        if beratKelinci[i] < beratMin {
            beratMin = beratKelinci[i]
        }
        if beratKelinci[i] > beratMax {
            beratMax = beratKelinci[i]
        }
    }

    return beratMin, beratMax
}
```

```

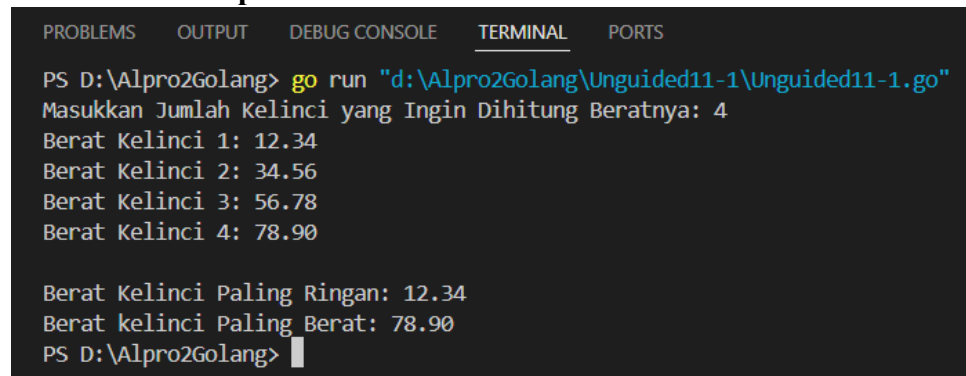
func main() {
    var n_226 int

    fmt.Print("Masukkan Jumlah Kelinci yang Ingin Dihitung
Beratnya: ")
    fmt.Scan(&n_226)

    beratKelinci := masukanBeratKelinci(n_226)
    beratMin, beratMax :=
cariBeratTerkecilTerbesar(beratKelinci)
    fmt.Printf("\nBerat Kelinci Paling Ringan: %.2f\nBerat
Kelinci Paling Berat: %.2f\n", beratMin, beratMax)
}

```

Screenshoot Output



```

PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Unguided11-1\Unguided11-1.go"
Masukkan Jumlah Kelinci yang Ingin Dihitung Beratnya: 4
Berat Kelinci 1: 12.34
Berat Kelinci 2: 34.56
Berat Kelinci 3: 56.78
Berat Kelinci 4: 78.90

Berat Kelinci Paling Ringan: 12.34
Berat kelinci Paling Berat: 78.90
PS D:\Alpro2Golang>

```

Deskripsi Program

Program ini merupakan implementasi untuk menghitung berat kelinci minimum dan maksimum dari sejumlah data berat kelinci yang dimasukkan oleh pengguna. Program memanfaatkan fungsi untuk memisahkan proses masukan data dan pencarian nilai minimum serta maksimum.

Fungsi **masukanBeratKelinci** digunakan untuk membaca data berat badan kelinci dari pengguna. Fungsi ini menerima parameter **n**, yaitu jumlah kelinci, lalu mengembalikan slice yang berisi berat badan kelinci setelah meminta input dari pengguna.

Fungsi **cariBeratTerkecilTerbesar** bertugas menentukan berat kelinci paling ringan (**beratMin**) dan paling berat (**beratMax**) dari data berat kelinci yang dimasukkan. Fungsi ini melakukan iterasi melalui slice

beratKelinci dan memperbarui nilai **beratMin** serta **beratMax** jika ditemukan nilai yang lebih kecil atau lebih besar selama iterasi.

Di dalam fungsi **main**, pengguna diminta untuk memasukkan jumlah kelinci yang datanya akan dihitung (**n_226**). Setelah itu, program memanggil fungsi **masukanBeratKelinci** untuk mengumpulkan data berat badan kelinci. Data ini kemudian diproses oleh fungsi **cariBeratTerkecilTerbesar** untuk menghitung nilai minimum dan maksimum berat kelinci.

2. Soal Studi Case

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 berat ikan-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

```
package main

import "fmt"

const maxIkan = 1000
var beratIkan [maxIkan]float64

func hitungTotalBerat(beratIkan []float64, x, y int)
([]float64, []int) {
    var totalBerat []float64
    var totalIkan []int

    for i := 0; i < x; i++ {
        if i%y == 0 {
```

```

        totalBerat = append(totalBerat, 0)
        totalIkan = append(totalIkan, 0)
    }
    totalBerat[len(totalBerat)-1] += beratIkan[i]
    totalIkan[len(totalIkan)-1]++
}
return totalBerat, totalIkan
}

func hitungRataRata(totalBerat []float64, totalIkan []int)
[]float64 {
    var rataRata []float64
    for i := 0; i < len(totalBerat); i++ {
        if totalIkan[i] > 0 {
            rataRata = append(rataRata,
totalBerat[i]/float64(totalIkan[i]))
        } else {
            rataRata = append(rataRata, 0)
        }
    }
    return rataRata
}

func main() {
    var x_226, y_226 int

    fmt.Print("Masukkan jumlah ikan dan ikan per wadah (x
y): ")
    fmt.Scan(&x_226, &y_226)
    fmt.Printf("Masukkan berat %d ikan: ", x_226)
    for i := 0; i < x_226; i++ {
        fmt.Scan(&beratIkan[i])
    }

    totalBerat, totalIkan :=
hitungTotalBerat(beratIkan[:x_226], x_226, y_226)

    fmt.Println("Total berat ikan di setiap wadah: ")
    for _, berat := range totalBerat {
        fmt.Printf("%.2f ", berat)
    }
    fmt.Println()

    fmt.Println("Rata-rata berat ikan di setiap wadah: ")
    rataRata := hitungRataRata(totalBerat, totalIkan)

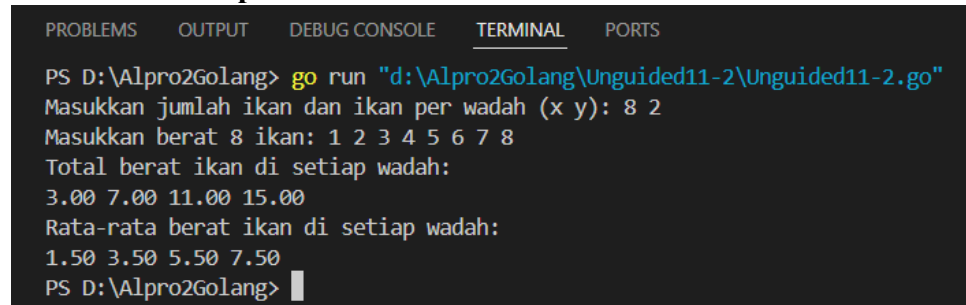
```

```

    for _, rata := range rataRata {
        fmt.Printf("%.2f ", rata)
    }
    fmt.Println()
}

```

Screenshoot Output



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Unguided11-2\Unguided11-2.go"
Masukkan jumlah ikan dan ikan per wadah (x y): 8 2
Masukkan berat 8 ikan: 1 2 3 4 5 6 7 8
Total berat ikan di setiap wadah:
3.00 7.00 11.00 15.00
Rata-rata berat ikan di setiap wadah:
1.50 3.50 5.50 7.50
PS D:\Alpro2Golang>

```

Deskripsi Program

Program ini merupakan implementasi untuk menghitung total berat ikan dan rata-rata berat ikan dalam sejumlah wadah, di mana setiap wadah memiliki kapasitas tertentu dalam jumlah ikan. Program menggunakan array **beratIkan** untuk menyimpan data berat masing-masing ikan dan memanfaatkan fungsi untuk menghitung total dan rata-rata berat berdasarkan pembagian ke dalam wadah.

Fungsi **hitungTotalBerat** bertugas menghitung total berat dan jumlah ikan di setiap wadah. Dengan parameter berupa array berat ikan, jumlah total ikan (**x**), dan kapasitas maksimal ikan per wadah (**y**). Hasilnya adalah dua array, yaitu **totalBerat** yang berisi total berat ikan di setiap wadah dan **totalIkan** yang berisi jumlah ikan dalam setiap wadah.

Fungsi **hitungRataRata** digunakan untuk menghitung rata-rata berat ikan di setiap wadah. Dengan parameter berupa array **totalBerat** dan **totalIkan**, fungsi ini membagi total berat ikan di setiap wadah dengan jumlah ikan dalam wadah tersebut untuk menghasilkan array rata-rata berat ikan.

Di dalam fungsi main, pengguna diminta untuk memasukkan jumlah ikan (**x_226**) dan kapasitas ikan per wadah (**y_226**). Setelah itu, program membaca berat masing-masing ikan dan menyimpannya dalam array **beratIkan**. Program kemudian memanggil fungsi **hitungTotalBerat** untuk menghitung total berat dan jumlah ikan per wadah, kemudian diikuti oleh fungsi **hitungRataRata** untuk menghitung rata-rata berat ikan di setiap wadah.

3. Soal Studi Case

Sebuah 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:

```
type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita; bMin, bMax *float64) {
  /* I.S. Terdefinisi array dinamis arrBerat

  Proses: Menghitung berat minimum dan maksimum dalam array
  F.S. Menampilkan berat minimum dan maksimum balita */
  ...
}

function rerata (arrBerat arrBalita) real {
  /* menghitung dan mengembalikan rerata berat balita dalam array */
  ...
}
```

Perhatikan sesi interaksi pada contoh berikut ini (teks bergaris bawah adalah input/read)

```
Masukan banyak data berat balita : 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
```

Sourcecode

```
package main

import "fmt"

type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, bmin, bmax *float64) {
  *bmin = arrBerat[0]
  *bmax = arrBerat[0]

  for i := 1; i < len(arrBerat); i++ {
    if arrBerat[i] != 0 {
```



```

        if arrBerat[i] < *bmin {
            *bmin = arrBerat[i]
        }
        if arrBerat[i] > *bmax {
            *bmax = arrBerat[i]
        }
    }
}

func ratarata(arrBerat arrBalita) float64 {
    var total float64
    var count int

    for i := 0; i < len(arrBerat); i++ {
        if arrBerat[i] != 0 {
            total += arrBerat[i]
            count++
        }
    }

    if count > 0 {
        return total / float64(count)
    }
    return 0
}

func main() {
    var n int
    var beratBalita arrBalita
    var min_226, max_226 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(&beratBalita[i])
    }

    hitungMinMax(beratBalita, &min_226, &max_226)

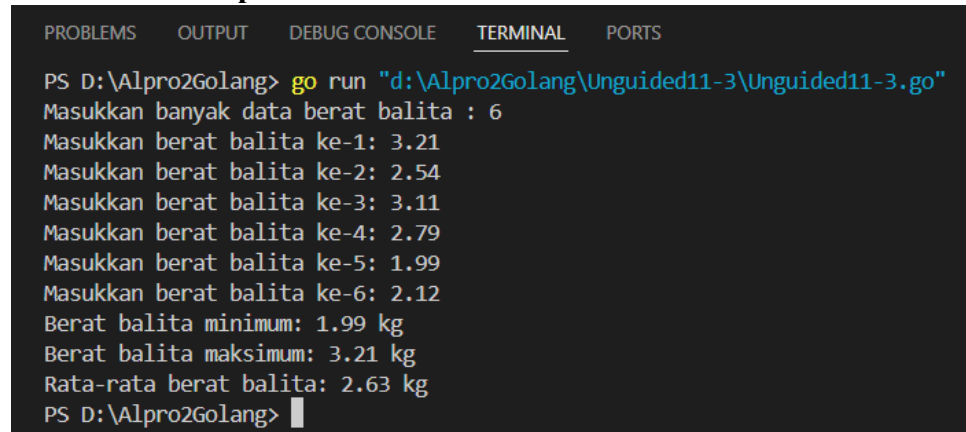
    rerata_226 := ratarata(beratBalita)

    fmt.Printf("Berat balita minimum: %.2f kg\n", min_226)
}

```

```
    fmt.Printf("Berat balita maksimum: %.2f kg\n", max_226)
    fmt.Printf("Rata-rata berat balita: %.2f kg\n",
rerata_226)
}
```

Screenshoot Output



The screenshot shows a terminal window with the following output:

```
PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Unguided11-3\Unguided11-3.go"
Masukkan banyak data berat balita : 6
Masukkan berat balita ke-1: 3.21
Masukkan berat balita ke-2: 2.54
Masukkan berat balita ke-3: 3.11
Masukkan berat balita ke-4: 2.79
Masukkan berat balita ke-5: 1.99
Masukkan berat balita ke-6: 2.12
Berat balita minimum: 1.99 kg
Berat balita maksimum: 3.21 kg
Rata-rata berat balita: 2.63 kg
PS D:\Alpro2Golang>
```

Deskripsi Program

Program ini merupakan implementasi untuk menghitung berat minimum, berat maksimum, dan rata-rata berat balita berdasarkan data yang dimasukkan oleh pengguna. Program menggunakan tipe data khusus **arrBalita** untuk mendefinisikan array dengan kapasitas maksimum 100 elemen yang menyimpan berat masing-masing balita.

Fungsi **hitungMinMax** bertugas menentukan berat minimum (**bmin**) dan maksimum (**bmax**) dari array **arrBerat**. Fungsi ini melakukan iterasi melalui elemen-elemen array, memeriksa setiap elemen, dan memperbarui nilai **bmin** atau **bmax** jika ditemukan nilai yang lebih kecil atau lebih besar. Fungsi ini juga mengabaikan elemen dengan nilai nol untuk memastikan hanya data valid yang diperhitungkan.

Fungsi **ratarata** digunakan untuk menghitung rata-rata berat balita. Fungsi ini menjumlahkan semua elemen valid dalam array **arrBerat** dan menghitung rata-rata dengan membagi total berat dengan jumlah elemen yang valid. Jika tidak ada data valid, fungsi akan mengembalikan nilai rata-rata 0.

Di dalam fungsi **main**, pengguna diminta untuk memasukkan jumlah balita yang datanya akan diolah. Program kemudian meminta pengguna untuk menginput berat masing-masing balita satu per satu, menyimpan data tersebut ke dalam array **beratBalita**. Setelah semua data dimasukkan,

program memanggil fungsi **hitungMinMax** untuk menentukan berat minimum dan maksimum, serta fungsi **ratarata** untuk menghitung rata-rata berat balita.