LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 11

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Disusun Oleh : Brian Farrel Evandhika

2311102037

IF 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

10.1 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:

I) 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.

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

maksimum:

Dalam Algoritma

```
max <- satu
i <- dua
sementara i kurang dari sama dengan n lakukan
  jika a[i] lebih besar dari a[max] maka
    max <- i
  endif
  i <- i ditambah satu
endwhile
```

Dalam notasi go

```
max = 0
i = 1
for i < n {
    if a[i] > a[max] {
        max = i
    }
    i = i + 1
}
```

10.2 Pencarlan NIIaI Ekstrlm pada Array Bertlpe 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!

```
type arrInt [2023]int


func terkecil_1(tabInt arrInt, n int) int {
   /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
   bilangan bulat */
   var min int = tabInt[0]  // min berisi data pertama
   var j int = 1            // pencarian dimulai dari data berikutnya
   for j < n {
      if min > tabInt[j] { // pengecekan apakah nilai minimum valid
         min = tabInt[j]  // update nilai minimum dengan yang valid
      }
      j = j + 1
   }
   return min  // returnkan nilai minimumnya
}
```

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada

bahasa Go di mulai dari nol atau "0" seperti penjelasan pada modul 9. Selanjutnya, 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. Oleh karena itu modifikasi pada program di atas dapat

dilihat pada potongan program berikut ini!

```
type arrInt [2023]int


func terkecil_2(tabInt arrInt, n int) int {
   /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi
   n bilangan bulat */
   var idx int = 0                                  // idx berisi indeks data pertama
   var j int = 1                                    // pencarian dimulai dari data berikutnya
   for j < n {
      if tabInt[idx] > tabInt[j] { // pengecekan apakah nilai minimum valid
         idx = j  // update nilai minimum dengan yang valid
      }
      j = j + 1
   }
   return idx  // returnkan indeks nilai minimumnya
}
```

10.3 Pencarlan NIIal Ekstrlm pada Array Bertlpe 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.

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

```
}


type arrMhs [2023]mahasiswa


func IPK_1(T arrMhs, n int) float64 {
    /* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
    n mahasiswa */
    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
}
```

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!

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

type arrMhs [2023]mahasiswa

func IPK_2(T arrMhs, n int) int {
    /* mengembalikan indeks mahasiswa yang memiliki ipk tertinggi pada array T yang
    berisi n mahasiswa */
    var idx int = 0
    var j int = 1
    for j < n {
        if T[idx].ipk < T[j].ipk {
            idx = j
        }
        j = j + 1
    }
    return idx
}
```

Sehingga melalui algoritma di atas, identitas mahasiswa dapat diperoleh, misalnya T[idx] . nama,

T[idx] . nim, T[idx] . kelas, hingga T[idx] .jurusan.

GUIDED

Soal Studi Case 1

TipeDasarIDX

Sourcecode

```
//2311102037_BRIAN FARREL EVANDHIKA_IF 11 06

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)
}
```

Screenshot Program

```
-Module PSReadLine'.

PS C:\Users\MSI GAMING> go run "c:\Users\MSI GAMING\Downloads\Modul 11\tempC
odeRunnerFile.go"
Masukkan jumlah elemen (maks 2023): 3
Masukkan elemen-elemen array:
Elemen ke-1: 1
Elemen ke-2: 2
Elemen ke-3: 3
Nilai terkecil dalam array adalah: 1 pada indeks: 0
PS C:\Users\MSI GAMING> go run "c:\Users\MSI GAMING\Downloads\Modul 11\tempC
odeRunnerFile.go"
Masukkan jumlah elemen (maks 2023): 3
Masukkan elemen-elemen array:
Elemen ke-1: 3
Elemen ke-2: 2
Elemen ke-3: 1
Nilai terkecil dalam array adalah: 1 pada indeks: 2
PS C:\Users\MSI GAMING>
```

Deskripsi Program

Program ini bertujuan untuk menemukan elemen terkecil dalam sebuah array dan menampilkan nilai serta indeks elemen tersebut. Program meminta pengguna untuk memasukkan jumlah elemen array, kemudian mengisi elemen-elemen tersebut. Setelah itu, program menggunakan fungsi terkecil untuk mencari indeks

elemen terkecil dalam array. Fungsi terkecil mengiterasi seluruh elemen dalam array, membandingkan setiap elemen untuk menemukan nilai terkecil, dan mengembalikan indeks elemen terkecil tersebut. Akhirnya, program menampilkan nilai dan indeks elemen terkecil yang ditemukan.

Soal Studi Case 2

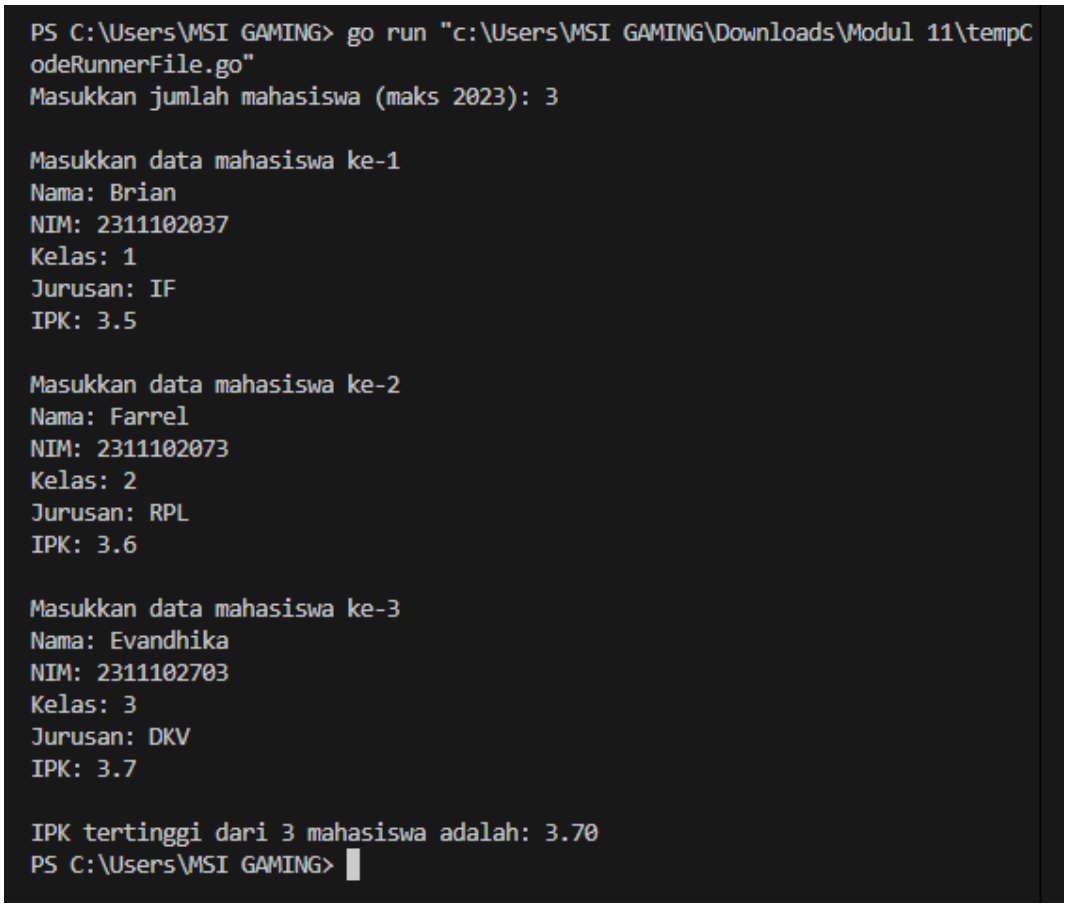
TipeStrukturIPK

Source Code

```
//2311102037_BRIAN FARREL EVANDHIKA_IF 11 06

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)


}
```

Screenshot Program



Deskripsi Program

Program ini mendata sejumlah mahasiswa dan mencari mahasiswa dengan IPK tertinggi. Program meminta input jumlah mahasiswa dan data masing-masing mahasiswa, seperti nama, NIM, kelas, jurusan, dan IPK. Setelah itu, program menggunakan fungsi ipk untuk mencari nilai IPK tertinggi dalam array mahasiswa, dan kemudian menampilkan IPK tertinggi tersebut. Dengan demikian, kita dapat mengetahui IPK tertinggi dari sekumpulan mahasiswa yang telah diinput.

II. UNGUIDED

Soal Studi Case 1

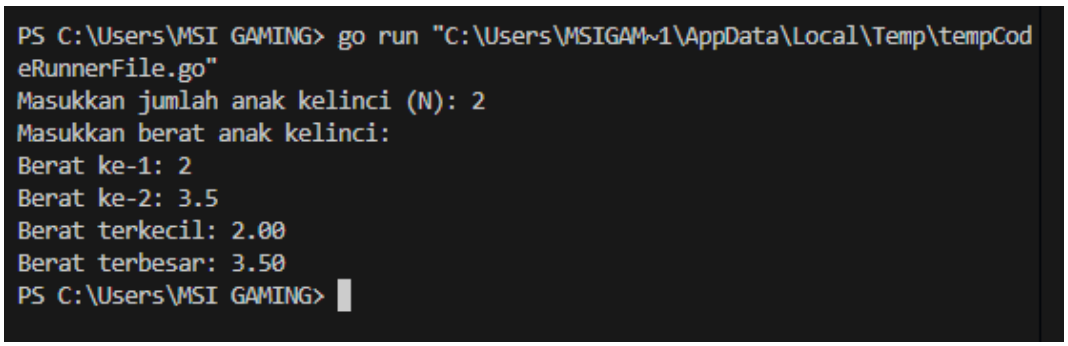
Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual ke pasar.
Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak
kelinci yang akan diJuaI.
Masukan terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat
N yang menyatakan 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.

Source Code

```go
//2311102037_BRIAN FARREL EVANDHIKA_IF 11 06
package main

import (
    "fmt"
    "math"
)

func main() {
    var N int
    var berat float64

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

    // Validasi kapasitas array
    if N <= 0 || N > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")
        return
    }

    beratKelinci := make([]float64, N)

    fmt.Println("Masukkan berat anak kelinci:")
    for i := 0; i < N; i++ {
```

```
        fmt.Printf("Berat ke-%d: ", i+1)
        fmt.Scan(&berat)
        beratKelinci[i] = berat
    }

    // Inisialisasi nilai terkecil dan terbesar
    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)
}
```

Screenshot Program



Deskripsi Program

Program ini bertujuan untuk membaca data berat anak kelinci, kemudian menentukan berat terkecil dan terbesar dari data tersebut. Pengguna diminta memasukkan jumlah anak kelinci (N), dan berat masing-masing anak kelinci. Program memvalidasi jumlah anak kelinci untuk memastikan nilainya antara 1 hingga 1000. Berat anak kelinci disimpan dalam array beratKelinci. Program kemudian menginisialisasi dua variabel untuk menyimpan berat terkecil (terkecil) dan terbesar (terbesar). Dengan mengiterasi melalui array beratKelinci, program

membandingkan setiap berat dengan nilai terkecil dan terbesar yang ada, lalu memperbarui nilai-nilai tersebut jika diperlukan. Akhirnya, program menampilkan berat terkecil dan terbesar dari anak-anak kelinci yang diinput.

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 dimasukan 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

dimasukan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah

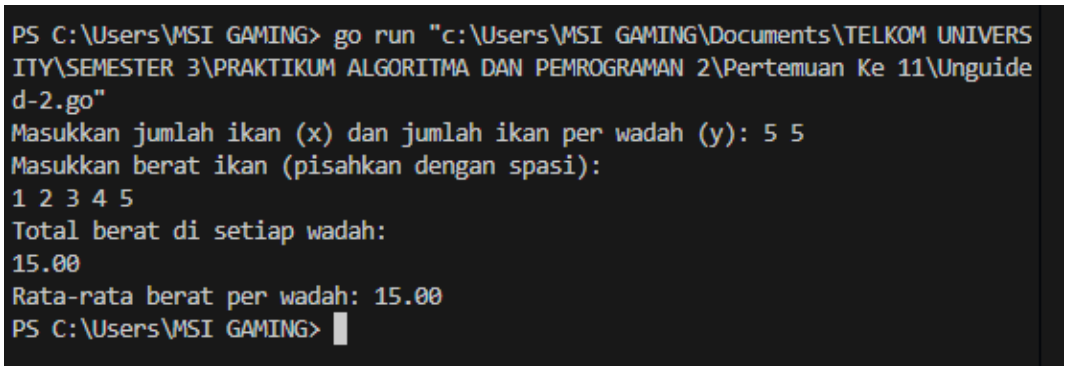
bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

Source Code

```go
// 2311102037_BRIAN FARREL EVANDHIKA_IF 11 06
package main

import (
    "fmt"
    "math"
)

func main() {
    // Masukan jumlah ikan (x) dan jumlah ikan per wadah (y)
    var x, y int
    fmt.Print("Masukkan jumlah ikan (x) dan jumlah ikan per wadah (y): ")
    fmt.Scan(&x, &y)

    // Validasi input
    if x <= 0 || y <= 0 || x > 1000 {
        fmt.Println("Input tidak valid. x harus > 0, y harus > 0, dan x <= 1000.")
        return
    }

    // Masukan berat ikan
    beratIkan := make([]float64, x)
    fmt.Println("Masukkan berat ikan (pisahkan dengan spasi):")
    for i := 0; i < x; i++ {
        fmt.Scan(&beratIkan[i])
    }

    // Menghitung total berat di setiap wadah
    jumlahWadah := int(math.Ceil(float64(x) / float64(y)))
    totalBeratPerWadah := make([]float64, jumlahWadah)
    for i := 0; i < x; i++ {
        indexWadah := i / y
        totalBeratPerWadah[indexWadah] += beratIkan[i]
    }

    // Menampilkan total berat setiap wadah
    fmt.Println("Total berat di setiap wadah:")
    for _, total := range totalBeratPerWadah {
        fmt.Printf("%.2f ", total)
    }
    fmt.Println()
```

```
    // Menghitung dan menampilkan rata-rata berat per wadah
    totalBeratKeseluruhan := 0.0
    for _, total := range totalBeratPerWadah {
        totalBeratKeseluruhan += total
    }
    rataRataBerat := totalBeratKeseluruhan / float64(jumlahWadah)
    fmt.Printf("Rata-rata berat per wadah: %.2f\n", rataRataBerat)
}
```

Screenshot Program



Deskripsi Program

Program ini digunakan untuk menghitung total berat ikan di setiap wadah dan rata-rata berat per wadah berdasarkan jumlah ikan dan kapasitas wadah yang diinput oleh pengguna. Pertama, program meminta pengguna untuk memasukkan jumlah ikan dan jumlah ikan per wadah. Kemudian, program meminta input berat masing-masing ikan. Setelah itu, program menghitung total berat ikan dalam setiap wadah dengan mengelompokkan ikan berdasarkan jumlah ikan per wadah dan menjumlahkan beratnya. Program menampilkan total berat ikan di setiap wadah dan menghitung rata-rata berat per wadah. Akhirnya, program menampilkan hasil perhitungan total berat setiap wadah dan rata-rata berat tersebut.

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.

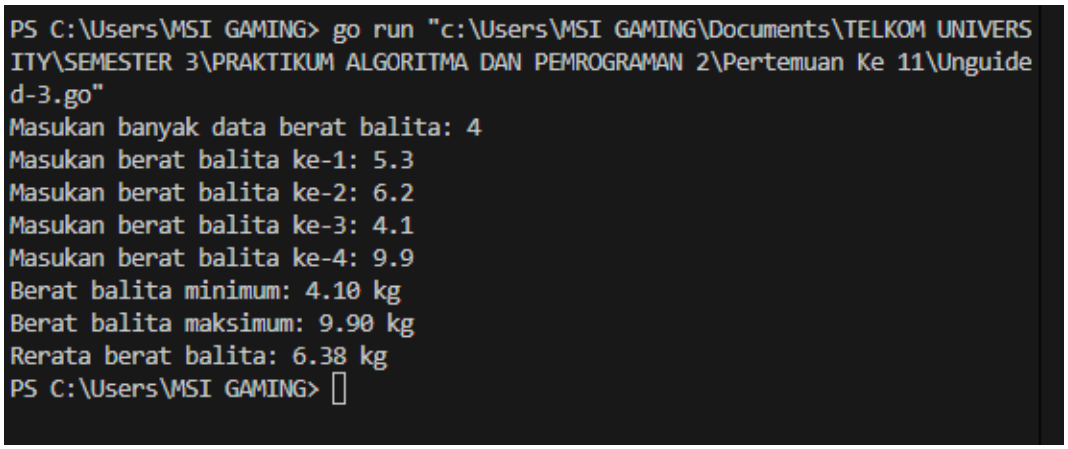
Buatlah program dengan spesifikasi subprogram sebagai berikut:

Source Code

```go
// 2311102037_BRIAN FARREL EVANDHIKA_IF 11 06
package main

import (
    "fmt"
)

type arrBalita [100]float64

// hitungMinMax menghitung berat minimum dan maksimum dalam array
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]
        }
    }
}

// rerata menghitung dan mengembalikan rata-rata berat balita dalam array
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

    // Input jumlah data balita
    fmt.Print("Masukan banyak data berat balita: ")
    fmt.Scan(&n)

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

    // Hitung minimum dan maksimum
    hitungMinMax(arrBerat, n, &bMin, &bMax)

    // Hitung rata-rata
    rata := rerata(arrBerat, n)

    // Output hasil
    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)
}
```

Screenshot Program



Deskripsi Program

Program ini bertujuan untuk menghitung berat minimum, maksimum, dan rata-rata dari sekumpulan data berat balita. Pengguna diminta memasukkan jumlah data balita dan berat masing-masing balita. Program menggunakan dua fungsi: hitungMinMax untuk menghitung berat minimum dan maksimum, serta rerata untuk menghitung rata-rata berat balita. Fungsi hitungMinMax membandingkan setiap berat balita untuk menemukan nilai minimum dan maksimum, sementara fungsi rerata menjumlahkan semua berat balita dan membaginya dengan jumlah balita untuk mendapatkan rata-rata. Akhirnya, program menampilkan hasil perhitungan berat minimum, maksimum, dan rata-rata tersebut.