

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2  
MODUL 10  
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



Oleh:

NAMA: Lutfi Shidqi Mardian

NIM: 103112400077

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO**

**2025**

## I. DASAR TEORI

### Tipe Bentuk (Struct dan Alias)

Dalam bahasa pemrograman Go (*Golang*), dikenal berbagai jenis **tipe bentuk** yang memungkinkan programmer untuk membuat struktur data kompleks dan lebih sesuai dengan kebutuhan aplikasi. Beberapa tipe bentuk yang sering digunakan dalam Go antara lain: **alias, struct, array, slice, dan map**. Berikut penjelasan masing-masing:

- **Alias (Type)**

Alias adalah pemberian nama baru untuk tipe data yang sudah ada, sehingga lebih mudah dibaca atau digunakan. Dalam Go, alias didefinisikan menggunakan kata kunci `type`.

Contoh:

```
type bilangan int  
  
type pecahan float64
```

Dengan cara ini, tipe `int` dan `float64` bisa diakses menggunakan nama baru yang lebih spesifik.

- **Struct (Structure/Record)**

Struct adalah kumpulan beberapa variabel yang memiliki hubungan, digabung menjadi satu kesatuan. Setiap elemen di dalam struct disebut `field`.

Contoh deklarasi struct di Go:

```
type waktu struct {  
    jam, menit, detik int  
}
```

Struct berguna untuk mengelompokkan data yang berkaitan, seperti data waktu, koordinat, atau informasi lainnya.

- **Array**

Array adalah kumpulan elemen dengan tipe data yang sama dan jumlah elemen yang tetap (statis) selama program berjalan. Di Go, deklarasi array menentukan jumlah elemen secara eksplisit.

Contoh deklarasi array:

```
var arr [5]int  
var buf = [5]byte{7, 3, 5, 2, 11}
```

Beberapa hal penting tentang array:

1. Indeks array di Go dimulai dari 0.
2. Ukuran array bisa diperiksa menggunakan fungsi `len(array)`.

3. Elemen dapat diakses dan dimodifikasi menggunakan indeks, misalnya `arr[0] = 10`.

- **Slice**

Slice adalah tipe data di Go yang mirip array tetapi memiliki ukuran yang bisa berubah selama eksekusi program. Slice lebih fleksibel dibandingkan array biasa. Slice bisa dibuat dari array, slice lain, atau menggunakan fungsi `make`.

Contoh deklarasi slice:

```
var s1 = []int{1, 2, 3, 4}

var s2 = make([]int, 10, 20)
```

Beberapa fungsi yang umum digunakan pada slice:

1. `len(slice)`: Mengembalikan jumlah elemen dalam slice.
2. `cap(slice)`: Mengembalikan kapasitas maksimum slice.
3. `append(slice, elemen)`: Menambahkan elemen baru ke slice

mfjmfj

- **Map**

Map adalah tipe data asosiatif di Go yang menyimpan pasangan key-value. Tidak seperti array, indeks pada map bisa berupa tipe data apapun (string, integer, float, dll).

Contoh deklarasi map:

```
var dct map[string]int

dct = map[string]int{"john": 25, "anne": 30}
```

## II. GUIDED

### 1.

```
package main

import "fmt"

func main() {

    var n int

    fmt.Print("Masukkan Jumlah Tanaman: ")

    fmt.Scan(&n)

    var tinggiTanaman [500]float64

    for i := 0; i < n; i++ {

        fmt.Printf("Masukkan tinggi tanaman ke-%d (cm): ", i+1)

        fmt.Scan(&tinggiTanaman[i])

    }

    min, max := tinggiTanaman[0], tinggiTanaman[0]

    for i := 1; i < n; i++ {

        if tinggiTanaman[i] < min {

            min = tinggiTanaman[i]

        }

        if tinggiTanaman[i] > max {

            max = tinggiTanaman[i]

        }

    }

    fmt.Printf("\nTinggi tanaman tertinggi: %.2f cm\n", max)

    fmt.Printf("\nTinggi tanaman terpendek: %.2f cm\n", min)

}
```

### Output Screenshot:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO02LUTFI> go run
Masukkan Jumlah Tanaman: 3
Masukkan tinggi tanaman ke-1 (cm): 12
Masukkan tinggi tanaman ke-2 (cm): 16
Masukkan tinggi tanaman ke-3 (cm): 18

Tinggi tanaman tertinggi: 18.00 cm

Tinggi tanaman terpendek: 12.00 cm
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO02LUTFI> |
```

### Penjelasan:

Program Go ini digunakan untuk mencatat dan menentukan tinggi tanaman tertinggi dan terpendek dari sejumlah tanaman yang dimasukkan oleh pengguna. Program dimulai dengan meminta input jumlah tanaman, kemudian pengguna diminta untuk memasukkan tinggi masing-masing tanaman dalam satuan sentimeter. Data tinggi tanaman disimpan dalam array, lalu program menghitung nilai tertinggi dan terendah dari data tersebut menggunakan perulangan. Hasil akhirnya, program akan menampilkan tinggi tanaman tertinggi dan terpendek dengan format dua angka di belakang koma.

## 2.

```
package main

import "fmt"

func main() {
    var x, y int
    rak: " "
    fmt.Print("Masukkan jumlah buku dan jumlah buku per
    fmt.Scan(&x,&y)
    var hargaBuku [500]float64
    Rp): "
    fmt.Println("\nMasukkan harga setiap buku (dalam ribuan
    for i := 0; i < x; i++ {
        fmt.Scan(&hargaBuku[i])
    }
    var hargaRataRata []float64
    for i := 0; i < x; i += y {
        total := 0.0
        for j := i; j < i+y && j < x; j++ {
            total += hargaBuku[j]
        }
        hargaRataRata = append(hargaRataRata,
total/float64(y))
    }

    min, max := hargaBuku[0], hargaBuku[0]
    for _, harga := range hargaBuku[:x] {
        if harga < min {
```

```

        min = harga
    }

    if harga > max {
        max = harga
    }
}

fmt.Printf("\nRata-rata harga per rak: ")
for _, avg := range hargaRataRata {
    fmt.Printf("%.2f", avg)
}

fmt.Printf("\nHarga Termahal: %.2f Rp\n", max)
fmt.Printf("Harga Termurah: %.2f Rp\n", min)
}

```

#### Output Screenshot:

```

● PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
Masukkan jumlah buku dan jumlah buku per rak: 2 4

Masukkan harga setiap buku (dalam ribuan Rp):
10000
20000

Rata-rata harga per rak: 7500.00
Harga Termahal: 20000.00 Rp
Harga Termurah: 10000.00 Rp
○ PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

**Penjelasan:**

Program Go ini berfungsi untuk menghitung rata-rata harga buku per rak serta menentukan harga buku termahal dan termurah dari sejumlah buku yang dimasukkan oleh pengguna. Pengguna diminta memasukkan jumlah total buku dan jumlah buku per rak, kemudian memasukkan harga masing-masing buku dalam satuan ribuan rupiah. Program menyimpan harga-harga tersebut dalam array, lalu menghitung rata-rata harga per rak dengan menjumlahkan harga buku dalam satu rak dan membaginya dengan jumlah buku per rak. Selain itu, program juga mencari dan menampilkan harga buku tertinggi dan terendah dari keseluruhan data. Hasil akhir ditampilkan dalam format dua angka di belakang koma.



### 3.

```
package main
import "fmt"
func main() {
    var n int
    fmt.Print("Masukkan jumlah siswa: ")
    fmt.Scan(&n)
    var nilaiSiswa [200]float64
    var totalNilai float64 = 0
    fmt.Println("\nMasukkan nilai ujian masing-masing
siswa: ")
    for i := 0; i < n; i++ {
        fmt.Scan(&nilaiSiswa[i])
        totalNilai += nilaiSiswa[i]
    }
    rataRata := totalNilai / float64(n)
    min,max := nilaiSiswa[0], nilaiSiswa[0]
    var diAtasRataRata int = 0
    for _, nilai := range nilaiSiswa[:n] {
        if nilai < min {
            min = nilai
        }
        if nilai > max {
            max = nilai
        }
        if nilai > rataRata {
            diAtasRataRata++
        }
    }
    fmt.Printf("\nNilai Terendah: %.0f\n", min)
    fmt.Printf("Nilai Tertinggi: %.0f\n", max)
    fmt.Printf("Rata-rata Kelas: %.0f\n", rataRata)
    fmt.Printf("Jumlah siswa di atas rata-rata: %d\n",
diAtasRataRata)
}
```

### Ss Output:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
Masukkan jumlah siswa: 5

Masukkan nilai ujian masing-masing siswa:
100
90
80
70
60

Nilai Terendah: 60
Nilai Tertinggi: 100
Rata-rata Kelas: 80
Jumlah siswa di atas rata-rata: 2
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

### Penjelasan:

Program Go ini digunakan untuk mengolah data nilai ujian siswa dalam suatu kelas. Program dimulai dengan meminta pengguna memasukkan jumlah siswa, lalu membaca nilai ujian masing-masing siswa dan menyimpannya dalam array. Nilai-nilai tersebut dijumlahkan untuk menghitung rata-rata kelas. Selanjutnya, program menentukan nilai tertinggi dan terendah serta menghitung jumlah siswa yang memperoleh nilai di atas rata-rata. Semua hasil, termasuk nilai minimum, maksimum, rata-rata kelas, dan jumlah siswa yang nilainya di atas rata-rata, ditampilkan dalam format yang ringkas dan jelas.

### III. UNGUIDED

#### 1.

```
package main

import "fmt"

func main() {

    var n int

    fmt.Print("Masukkan Jumlah Kelinci: ")

    fmt.Scan(&n)

    var beratKenlinci [100]float64

    for i := 0; i < n; i++ {

        i+1)      fmt.Printf("Masukkan berat kelinci ke-%d (Kg): ",

        fmt.Scan(&beratKenlinci[i])

    }

    min, max := beratKenlinci[0], beratKenlinci[0]

    for i := 1; i < n; i++ {

        if beratKenlinci[i] < min {

            min = beratKenlinci[i]

        }

        if beratKenlinci[i] > max {

            max = beratKenlinci[i]

        }

    }

    fmt.Printf("\nBerat kelinci terkecil: %.2f kg\n", max)

    fmt.Printf("\nBerat kelinci terbesar: %.2f kg\n", min)

}
```

### Output Screenshot:

```
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
Masukkan Jumlah Kelinci: 3
Masukkan berat kelinci ke-1 (Kg): 1.2
Masukkan berat kelinci ke-2 (Kg): 2.3
Masukkan berat kelinci ke-3 (Kg): 1.7

Berat kelinci terkecil: 2.30 kg

Berat kelinci terbesar: 1.20 kg
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>
```

### Penjelasan:

Program Go ini digunakan untuk mencatat dan menentukan berat kelinci terbesar dan terkecil dari sejumlah kelinci yang dimasukkan oleh pengguna. Program meminta input jumlah kelinci, kemudian berat masing-masing kelinci dalam satuan kilogram dimasukkan dan disimpan dalam array. Setelah semua data dimasukkan, program membandingkan setiap nilai untuk mencari berat minimum dan maksimum. Namun, terdapat kesalahan dalam output: label “terkecil” dan “terbesar” tertukar—yang ditampilkan sebagai “terkecil” justru adalah nilai maksimum, dan sebaliknya. Setelah diperbaiki, program akan menampilkan berat kelinci terbesar dan terkecil dengan format dua angka di belakang koma.

## 2.

```
package main

import "fmt"

func main() {

    var x, y int

    fmt.Print("Masukkan banyak ikan dan jumlah per wadah: ")

    fmt.Scan(&x, &y)

    var beratIkan [1000]float64

    fmt.Println("Masukkan berat ikan:")

    for i := 0; i < x; i++ {

        fmt.Scan(&beratIkan[i])

    }

    totalWadah := x / y

    if x%y != 0 {

        totalWadah++

    }

    var totalPerWadah [1000]float64

    idx := 0

    for i := 0; i < totalWadah; i++ {

        var total float64 = 0

        for j := 0; j < y && idx < x; j++ {

            total += beratIkan[idx]

            idx++

        }

    }

}
```

```

    }

    totalPerWadah[i] = total

}

fmt.Println("Total berat per wadah:")

for i := 0; i < totalWadah; i++ {

    fmt.Printf("%.2f ", totalPerWadah[i])

}

fmt.Println()

var totalSemua float64 = 0

for i := 0; i < totalWadah; i++ {

    totalSemua += totalPerWadah[i]

}

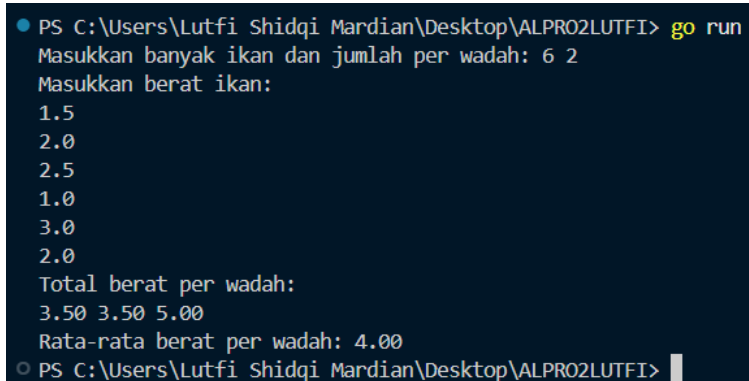
rataRata := totalSemua / float64(totalWadah)

fmt.Printf("Rata-rata berat per wadah: %.2f\n", rataRata)

}

```

### Output Screenshot:



```

PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
Masukkan banyak ikan dan jumlah per wadah: 6 2
Masukkan berat ikan:
1.5
2.0
2.5
1.0
3.0
2.0
Total berat per wadah:
3.50 3.50 5.00
Rata-rata berat per wadah: 4.00
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

**Penjelasan:**

Program Go ini digunakan untuk menghitung total dan rata-rata berat ikan per wadah berdasarkan jumlah ikan dan kapasitas tiap wadah yang dimasukkan oleh pengguna. Setelah pengguna memasukkan jumlah total ikan dan jumlah ikan per wadah, program menerima input berat masing-masing ikan dalam kilogram. Data berat ikan disimpan dalam array, kemudian dikelompokkan ke dalam beberapa wadah. Program menghitung total berat ikan di setiap wadah dan menyimpannya dalam array lain. Setelah itu, ditampilkan total berat untuk setiap wadah serta rata-rata berat antar seluruh wadah dengan dua angka di belakang koma. Program ini berguna untuk mendistribusikan berat secara merata saat pengemasan ikan.

### 3.

```
papackage main

import "fmt"

type arrBalita [100]float64

func hitungMinMax(arr arrBalita, n int, bMin *float64, bMax *float64) {

    if n == 0 {

        return

    }

    *bMin = arr[0]

    *bMax = arr[0]

    for i := 1; i < n; i++ {

        if arr[i] < *bMin {

            *bMin = arr[i]

        }

        if arr[i] > *bMax {

            *bMax = arr[i]

        }

    }

}

func r(arr arrBalita, n int) float64 {

    var total float64

    for i := 0; i < n; i++ {

        total += arr[i]

    }

    return total / float64(n)

}

func main() {
```



```

var data arrBalita

var n int

var min, max float64

fmt.Print("Masukan banyak data berat balita: ")

fmt.Scan(&n)

for i := 0; i < n; i++ {

    fmt.Printf("Masukan berat balita ke-%d: ", i+1)

    fmt.Scan(&data[i])

}

hitungMinMax(data, n, &min, &max)

rBerat := r(data, n)

fmt.Printf("\nBerat balita minimum: %.2f kg\n", min)

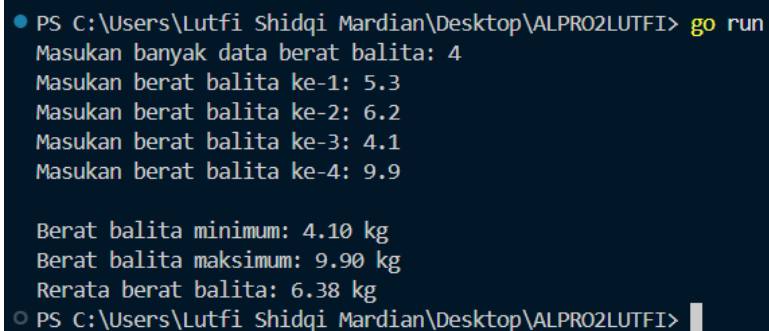
fmt.Printf("Berat balita maksimum: %.2f kg\n", max)

fmt.Printf("Rerata berat balita: %.2f kg\n", rBerat)

}

```

### Output Screenshot:



```

PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI> go run
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
PS C:\Users\Lutfi Shidqi Mardian\Desktop\ALPRO2LUTFI>

```

**Penjelasan:**

Program Go ini dirancang untuk mengolah data berat badan balita, dengan menghitung nilai minimum, maksimum, dan rata-ratanya. Data berat balita disimpan dalam array bertipe khusus `arrBalita`, dan jumlah data ditentukan oleh input pengguna. Program menggunakan dua fungsi terpisah: `hitungMinMax` untuk menentukan berat minimum dan maksimum dengan bantuan pointer, serta `r` untuk menghitung rata-rata berat badan. Setelah semua data dimasukkan, hasil perhitungan ditampilkan dalam satuan kilogram dengan dua angka di belakang koma. Program ini cocok digunakan untuk analisis sederhana data pertumbuhan balita.

#### IV. KESIMPULAN

Dari implementasi berbagai konsep seperti tipe bentukan, array, slice, map, pointer, dan fungsi dalam praktikum ini, dapat disimpulkan bahwa penggunaan struktur data yang tepat dalam bahasa Go sangat berperan dalam menyusun program yang rapi, efisien, dan mudah dipahami. Tipe bentukan seperti struct dan alias membantu memodelkan data kompleks secara lebih jelas, sementara array dan slice memfasilitasi pengelolaan data yang terstruktur dan dinamis. Map mempermudah pencatatan data yang membutuhkan pasangan kunci dan nilai, sedangkan pointer memungkinkan manipulasi data secara langsung melalui referensi memori. Selain itu, pencarian nilai ekstrem dalam array menunjukkan pentingnya logika algoritmik dalam menyaring informasi penting dari kumpulan data. Dengan memahami dan menguasai konsep-konsep ini, mahasiswa dapat menyusun program yang modular, mudah dikembangkan, dan siap menangani permasalahan pemrosesan data yang lebih kompleks di dunia nyata.

#### V. REFERENSI

1. Donovan, A. A., & Kernighan, B. W. (2015). *The Go Programming Language*. Addison-Wesley.
2. Official Golang Documentation. (n.d.). *Effective Go - Structs*. Retrieved from [https://go.dev/doc/effective\\_go#structs](https://go.dev/doc/effective_go#structs)
3. Official Golang Documentation. (n.d.). *Effective Go - Arrays and Slices*. Retrieved from [https://go.dev/doc/effective\\_go#arrays](https://go.dev/doc/effective_go#arrays)
4. W3Schools. (n.d.). *Golang Arrays and Slices Tutorial*. Retrieved from [https://www.w3schools.com/go/go\\_arrays.php](https://www.w3schools.com/go/go_arrays.php)
5. GeeksforGeeks. (n.d.). *Pointers in Golang*. Retrieved from <https://www.geeksforgeeks.org/pointers-in-golang/>
6. Go.dev Blog. (n.d.). *Slices: usage and internals*. Retrieved from <https://go.dev/blog/slices-intro>
7. Ardan Labs. (n.d.). *Understanding Go Pointers*. Retrieved from <https://www.ardanlabs.com/blog/2013/05/memcpy-and-pointers-in-go.html>
8. GeeksforGeeks. (n.d.). *Finding Minimum and Maximum in an Array*. Retrieved from <https://www.geeksforgeeks.org/c-program-find-largest-element-array/>