LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 10

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

NAMA: ICHYA ULUMIDDIIN

NIM: 103112400076

KELAS: 12IF-01-A

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Dalam pengolahan data, penting untuk mengetahui nilai ekstrem, yaitu nilai minimum dan maksimum, dari suatu himpunan data. Nilai minimum adalah nilai terkecil, sedangkan nilai maksimum adalah nilai terbesar dalam kumpulan data tersebut. Mengetahui nilai ekstrem membantu dalam analisis data, seperti menentukan rentang nilai, mendeteksi outlier, dan memahami distribusi data.

1. **Menggunakan Perulangan Manual:**

   Cara tradisional untuk menemukan nilai minimum dan maksimum adalah dengan menginisialisasi variabel `min` dan `max` dengan nilai pertama dari array atau slice, kemudian mengiterasi elemen-elemen berikutnya untuk memperbarui `min` dan `max` sesuai kebutuhan.

2. **Array dengan Tipe Data Dasar**

   Dalam Golang, array dengan tipe data dasar seperti int, float64, atau string dapat digunakan untuk menyimpan sekumpulan data homogen. Untuk menemukan nilai minimum dan maksimum dalam array tersebut, kita dapat menggunakan perulangan for untuk membandingkan setiap elemen.

3. **Array dengan Data Terstruktur (Struct)**

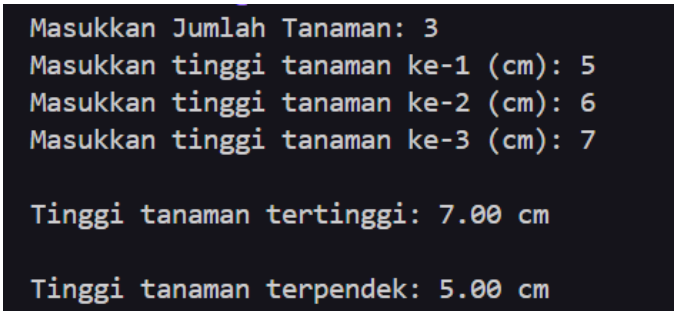
   Golang memungkinkan kita untuk mendefinisikan struktur data menggunakan struct. Misalnya, kita memiliki struktur Mahasiswa dengan atribut Nama dan Nilai. Untuk mencari mahasiswa dengan nilai tertinggi dan terendah, kita dapat melakukan iterasi pada slice dari Mahasiswa.

II. GUIDED

Source Code 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



Penjelasan

Program ini bertujuan untuk mencatat tinggi beberapa tanaman, lalu menentukan tinggi tanaman tertinggi dan terpendek dari data yang dimasukkan. Pertama, pengguna diminta untuk memasukkan jumlah tanaman yang akan dicatat. Selanjutnya, pengguna diminta untuk mengisi tinggi dari masing-masing tanaman satu per satu, yang akan disimpan dalam array bertipe float64 dengan kapasitas maksimal 500 elemen.

Setelah semua data tinggi tanaman dimasukkan, program melakukan pencarian nilai maksimum dan minimum dengan cara membandingkan satu per satu nilai dalam array. Nilai awal untuk tinggi tertinggi dan terpendek diambil dari elemen pertama, lalu diperbarui jika ditemukan data yang lebih tinggi atau lebih rendah. Hasil akhirnya adalah tampilan tinggi tanaman tertinggi dan terpendek, yang dicetak dalam dua angka desimal untuk ketelitian.

Source Code Guided 2

```
package main

import "fmt"

func main() {
	var x, y int
	fmt.Print("Masukkan jumlah buku dan jumlah buku per rak:
")
	fmt.Scan(&x,&y)

	var hargaBuku [500]float64
	fmt.Println("\nMasukkan harga setiap buku (dalam ribuan
Rp): ")
	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

```
Masukkan jumlah buku dan jumlah buku per rak: 3 4

Masukkan harga setiap buku (dalam ribuan Rp):
30
900
76000

Rata-rata harga per rak: 19232.50
Harga Termahal: 76000.00 Rp
Harga Termurah: 30.00 Rp
```

Penjelasan

Program ini digunakan untuk mengelola dan menganalisis data harga buku dalam konteks penyusunan rak buku. Program dimulai dengan meminta input dari pengguna berupa jumlah total buku (x) dan jumlah buku yang akan disusun dalam satu rak (y). Selanjutnya, pengguna diminta untuk memasukkan harga dari setiap buku (dalam ribuan rupiah), yang disimpan dalam array hargaBuku berkapasitas 500 elemen.

Setelah semua data harga buku dimasukkan, program menghitung rata-rata harga per rak. Perhitungan dilakukan dengan cara menjumlahkan harga beberapa buku sesuai kapasitas y per rak, lalu dibagi dengan jumlah tersebut. Hasilnya disimpan dalam slice hargaRataRata. Program juga mencari harga buku tertinggi (termahal) dan terendah (termurah) dari seluruh data yang dimasukkan.

Terakhir, program menampilkan rata-rata harga untuk setiap rak secara berurutan, serta mencetak harga buku paling mahal dan paling murah.

Source Code Guided 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)
}
```

Output

```
Masukkan jumlah siswa: 3

Masukkan nilai ujian masing-masing siswa:
90
89
96

Nilai Terendah: 89
Nilai Tertinggi: 96
Rata-rata Kelas: 92
Jumlah siswa di atas rata-rata: 1
```

Penjelasan

Program ini bertujuan untuk mencatat dan menganalisis nilai ujian siswa dalam sebuah kelas. Pertama, pengguna akan diminta memasukkan jumlah siswa, kemudian dilanjutkan dengan memasukkan nilai ujian untuk setiap siswa. Nilai-nilai ini disimpan dalam array nilaiSiswa dengan kapasitas maksimum 200 elemen, dan juga langsung dijumlahkan ke variabel totalNilai untuk keperluan menghitung rata-rata.

Setelah semua data dimasukkan, program menghitung rata-rata nilai kelas dengan membagi total nilai dengan jumlah siswa. Selanjutnya, program mencari nilai tertinggi (maksimum) dan nilai terendah (minimum) dengan membandingkan tiap elemen dalam array. Selain itu, program juga menghitung berapa banyak siswa yang memiliki nilai di atas rata-rata. Akhirnya, program akan menampilkan hasil analisis berupa nilai terendah, nilai tertinggi, rata-rata kelas, dan jumlah siswa yang nilainya di atas rata-rata.

III. UNGUIDED

Source Code Unguided 1

```
package main

import (
	"fmt"
)

func main() {
	var N int
	var berat [1000]float64
	fmt.Print("Masukkan jumlah anak kelinci: ")
	fmt.Scan(&N)
	if N > 1000 || N <= 0 {
		fmt.Println("Jumlah anak kelinci harus antara
1 hingga 1000.")
		return
	}
	fmt.Println("Masukkan berat anak kelinci:")
	for i := 0; i < N; i++ {
		fmt.Scan(&berat[i])
	}
	minBerat := berat[0]
	maxBerat := berat[0]
	for i := 1; i < N; i++ {
		if berat[i] < minBerat {
			minBerat = berat[i]
		}
		if berat[i] > maxBerat {
			maxBerat = berat[i]
		}
	}
	fmt.Printf("Berat terkecil: %.2f\n", minBerat)
	fmt.Printf("Berat terbesar: %.2f\n", maxBerat)
}
```

Output

```
Masukkan jumlah anak kelinci: 3
Masukkan berat anak kelinci:
39
23
21
Berat terkecil: 21.00
Berat terbesar: 39.00
```

Penjelasan

Program ini dibuat untuk mencatat dan menganalisis berat anak kelinci, dengan tujuan utama mencari berat terkecil dan berat terbesar dari data yang dimasukkan oleh pengguna. Pertama, pengguna diminta untuk memasukkan jumlah anak kelinci, kemudian mengisi berat masing-masing kelinci. Data berat tersebut disimpan dalam array berat dengan kapasitas maksimal 1000 elemen.

Setelah semua data dimasukkan, program melakukan perbandingan antar elemen untuk menentukan nilai minimum dan maksimum dari seluruh berat kelinci yang tersedia. Nilai awal minimum dan maksimum diambil dari elemen pertama array, kemudian diperbarui saat ditemukan nilai yang lebih kecil atau lebih besar. Terakhir, program menampilkan hasil berupa berat terkecil dan terbesar, masing-masing ditampilkan dengan dua angka di belakang koma.

Source Code Unguided 2

```
package main

import (
	"fmt"
)

func main() {
	var x, y int
	var berat [1000]float64
	fmt.Print("Masukkan jumlah ikan dan jumlah ikan per
wadah (x y): ")
	fmt.Scan(&x, &y)

	if x > 1000 || x <= 0 || y <= 0 {
		fmt.Println("Nilai x harus 1-1000 dan y > 0")
		return
	}
	fmt.Println("Masukkan berat ikan:")
	for i := 0; i < x; i++ {
		fmt.Scan(&berat[i])
	}
	jumlahWadah := (x + y - 1) / y
	totalWadah := make([]float64, jumlahWadah)
	totalBeratSemua := 0.0

	for i := 0; i < x; i++ {
		idx := i / y
		totalWadah[idx] += berat[i]
		totalBeratSemua += berat[i]
	}
	fmt.Println("Total berat tiap wadah:")
	for _, total := range totalWadah {
		fmt.Printf("%.2f ", total)
	}
	fmt.Println()
	rataRata := totalBeratSemua / float64(jumlahWadah)
	fmt.Printf("Berat rata-rata per wadah: %.2f\n",
rataRata)
}
```

Output

```
Masukkan jumlah ikan dan jumlah ikan per wadah (x y): 6 3
Masukkan berat ikan:
12
10
9
3
23
37
Total berat tiap wadah:
31.00 63.00
Berat rata-rata per wadah: 47.00
```

Penjelasan

Program ini dirancang untuk membantu petugas dalam mencatat dan menghitung berat ikan yang akan dimasukkan ke dalam beberapa wadah sebelum dijual atau didistribusikan. Pada awalnya, program meminta pengguna untuk memasukkan dua angka: jumlah total ikan (x) dan jumlah ikan per wadah (y). Kemudian, pengguna diminta untuk menginput berat dari masing-masing ikan, yang disimpan dalam array berat berkapasitas 1000 elemen.

Setelah semua data berat ikan dimasukkan, program mulai membagi ikan ke dalam beberapa wadah secara berurutan berdasarkan jumlah y per wadah. Total berat dari setiap wadah disimpan dalam slice totalWadah. Perhitungan jumlah wadah juga sudah disesuaikan agar tetap akurat walaupun jumlah ikan tidak habis dibagi rata. Program juga menjumlahkan seluruh berat ikan ke dalam totalBeratSemua untuk nantinya digunakan dalam menghitung rata-rata berat per wadah.

Hasil akhir yang ditampilkan meliputi total berat di tiap wadah dan berat rata-rata per wadah, dengan masing-masing nilai ditampilkan dalam dua angka di belakang koma.

Source Code Unguided 3

```
package main

import (
	"fmt"
)

type arrBalita [100]float64
func hitungMinMax(arrBerat arrBalita, jumlah int, bMin, bMax
*float64) {
	*bMin = arrBerat[0]
	*bMax = arrBerat[0]
	for i := 1; i < jumlah; i++ {
		if arrBerat[i] < *bMin {
			*bMin = arrBerat[i]
		}
		if arrBerat[i] > *bMax {
			*bMax = arrBerat[i]
		}
	}
}
func rerata(arrBerat arrBalita, jumlah int) float64 {
	var total float64 = 0.0
	for i := 0; i < jumlah; i++ {
		total += arrBerat[i]
	}
	return total / float64(jumlah)
}
func main() {
	var dataBerat arrBalita
	var n int

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

	if n <= 0 || n > 100 {
		fmt.Println("Jumlah balita harus antara 1
sampai 100.")
		return
	}
	for i := 0; i < n; i++ {
		fmt.Printf("Masukan berat balita ke-%d: ", i+1)
		fmt.Scan(&dataBerat[i])
	}
	var beratMin, beratMax float64
	hitungMinMax(dataBerat, n, &beratMin, &beratMax)
	rerataBerat := rerata(dataBerat, n)
	fmt.Printf("Berat balita minimum: %.2f kg\n",
beratMin)
	fmt.Printf("Berat balita maksimum: %.2f kg\n",
beratMax)
	fmt.Printf("Rerata berat balita: %.2f kg\n",
rerataBerat)
}
```

Output

```
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
```

Penjelasan

Program ini dibuat untuk membantu petugas Posyandu dalam mencatat dan menganalisis data berat balita. Tujuan utamanya adalah mencari berat minimum, berat maksimum, dan rerata berat balita dari sejumlah data yang dimasukkan. Data berat balita disimpan dalam array khusus bertipe arrBalita yang dapat menampung hingga 100 data bertipe float64.
Pertama, pengguna akan diminta untuk memasukkan jumlah balita yang datanya ingin dicatat, lalu memasukkan berat masing-masing balita. Program kemudian memanggil subprogram hitungMinMax yang berfungsi untuk mencari nilai berat terkecil dan terbesar dari array. Nilai awal diambil dari data pertama, lalu dibandingkan satu per satu untuk menemukan nilai ekstrem.
Selain itu, fungsi rerata digunakan untuk menghitung rata-rata berat balita dengan menjumlahkan seluruh nilai dalam array lalu membaginya dengan jumlah balita. Hasil akhirnya berupa tampilan tiga informasi penting: berat minimum, berat maksimum, dan rerata berat, semuanya ditampilkan dalam format dua angka di belakang koma.

IV. KESIMPULAN

Pencarian nilai maksimum dan minimum merupakan bagian penting dalam pengolahan data, terutama ketika kita ingin mengetahui sebaran atau ekstrem dari suatu kumpulan nilai. Dalam konteks bahasa pemrograman Go (Golang), proses ini dapat dilakukan dengan cara sederhana menggunakan struktur data array dan perulangan. Dengan membandingkan setiap elemen satu per satu, program dapat menentukan mana nilai terbesar (maksimum) dan terkecil (minimum) dalam sebuah dataset, baik itu berupa berat balita, tinggi tanaman, harga barang, atau nilai ujian. Pendekatan ini efektif karena bekerja dengan kompleksitas waktu linier, yaitu O(n), di mana n adalah jumlah data.

Melalui penggunaan fungsi atau subprogram seperti hitungMinMax, kode menjadi lebih terstruktur dan mudah dipelihara. Selain memberikan hasil yang akurat, pemisahan logika pencarian nilai ekstrem ke dalam fungsi tersendiri juga membantu meningkatkan keterbacaan program dan memudahkan penggunaan kembali dalam berbagai konteks.

.

REFERENSI

A., F. (2025, January 8). *Apa Itu Struktur Data? Pahami Arti, Jenis, dan Fungsinya dalam Pemrograman*. Retrieved from HOSTINGER TUTORIAL: https://www.hostinger.com/id/tutorial/apa-itu-struktur-data

Fazry. (2024, May 25). *Array: Konsep, Implementasi, dan Penggunaan*. Retrieved from Rumah Coding: https://rumahcoding.co.id/array-konsep-implementasi-dan-penggunaan/