LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 7

MATERI

Oleh:

NAMA: Muhammad Fahruli Ma’ruf

NIM: 103112400057

KELAS: 12-IF-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

Koleksi kode Go ini memberikan gambaran menyeluruh tentang **prinsip-prinsip dasar pemrograman Go**. Intinya, kode-kode ini membahas:

- **Arsitektur Program Go**: Struktur dasar setiap aplikasi Go dimulai dengan package main dan func main().
- **Pengelolaan Data**: Implementasi berbagai tipe data, termasuk primitif, serta tipe kustom seperti alias dan struct untuk pengelompokan data logis. Program secara ekstensif menggunakan **array** (koleksi ukuran tetap) dan **slice** (koleksi dinamis), mendemonstrasikan operasi seperti penambahan, penghapusan, pencarian, dan analisis statistik (rata-rata, standar deviasi).
- **Modularitas Fungsional**: Penggunaan **fungsi** (blok kode yang mengembalikan nilai) dan **prosedur** (blok kode yang melakukan tugas tanpa mengembalikan nilai) untuk membuat kode lebih terstruktur dan mudah dikelola.
- **Pendekatan Algoritmik**: Perbandingan dan penerapan dua metode utama:
    - **Rekursi**: Teknik di mana fungsi memanggil dirinya sendiri untuk memecahkan masalah dengan membaginya menjadi sub-masalah yang lebih kecil. Ini memerlukan base case (kondisi berhenti) dan recursive case (kondisi pemanggilan ulang).
    - **Iterasi**: Menggunakan perulangan untuk tugas-tugas berulang seperti perhitungan faktorial, perpangkatan, deret Fibonacci, dan pola visual.
- **Aplikasi Praktis**: Kode ini menyimulasikan beragam skenario nyata seperti pencatatan hasil pertandingan sepak bola, manajemen nilai mahasiswa, penentuan pemenang kompetisi berdasarkan skor, pengecekan palindrom, serta perhitungan dasar seperti gaji lembur dan status kelulusan.

II. GUIDED

```go
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[1] < min {
            min = tinggiTanaman[i]
        }
        if tinggiTanaman[1] > max {
            max = tinggiTanaman[i]
        }
    }

    fmt.Printf("\nTinggi tanaman tertinggi: %.2f cm\n", max)
    fmt.Printf("Tinggi tanaman terpendek: %.2f cm\n", min)
}
```

Penjelasan: Inti dari semua kode Go yang Anda berikan adalah demonstrasi komprehensif tentang dasar-dasar pemrograman Go. Program-program ini secara kolektif menampilkan:

1. **Struktur dan Tipe Data**: Memperkenalkan fondasi program Go (package main, func main()), serta penggunaan tipe data dasar dan kustom (struct, alias), termasuk array (tetap) dan slice (dinamis) dengan operasi CRUD (Create, Read, Update, Delete) serta statistik.

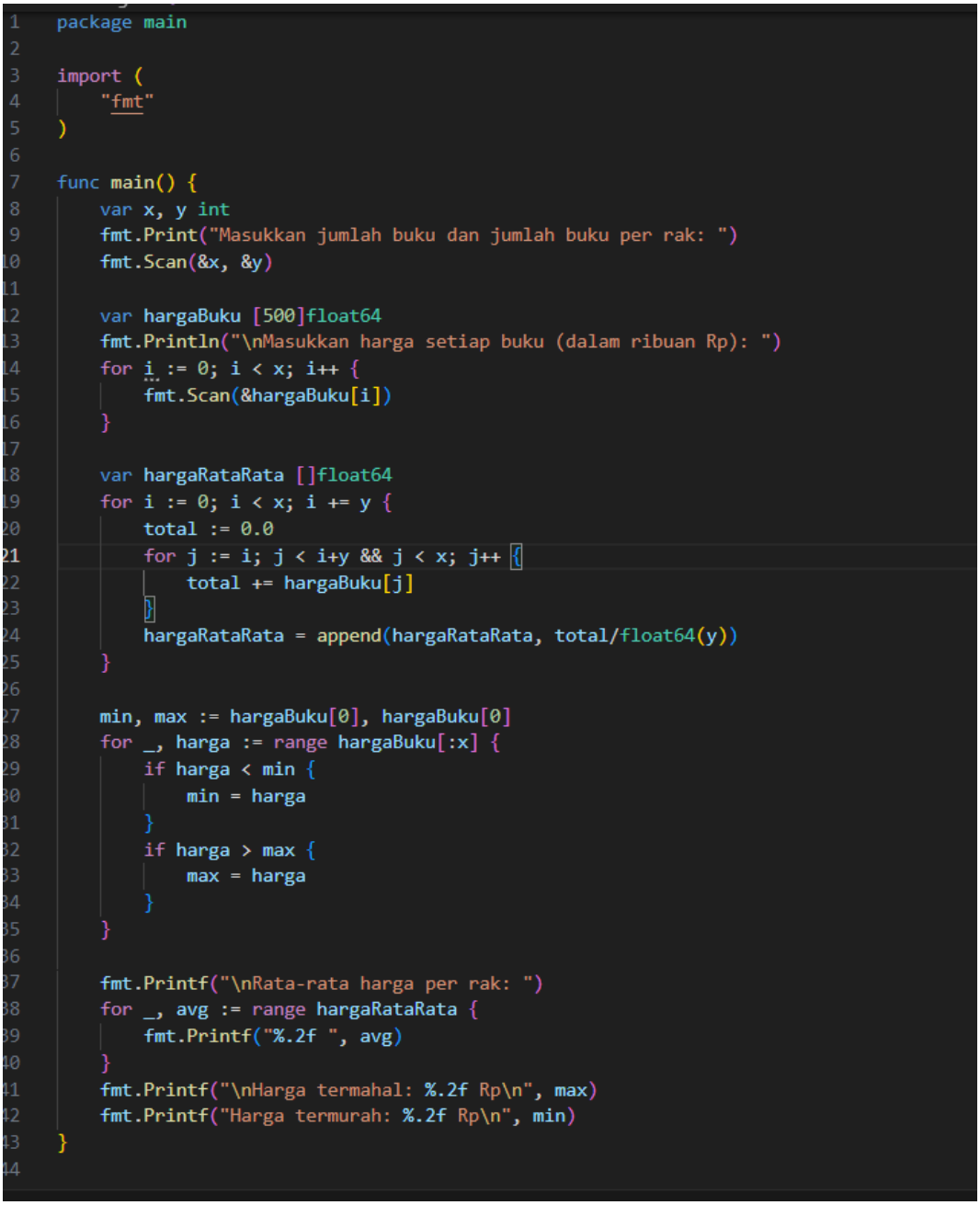
2. **Modularitas Kode**: Menjelaskan konsep fungsi (subprogram yang mengembalikan nilai) dan prosedur (subprogram yang melakukan tugas tanpa mengembalikan nilai) untuk menciptakan kode yang terstruktur dan mudah dikelola.

3. **Implementasi Algoritma**: Membandingkan dan menerapkan dua pendekatan utama: **rekursi** (fungsi memanggil dirinya sendiri dengan base case dan recursive case sebagai inti) dan **iterasi** (menggunakan perulangan), untuk memecahkan masalah matematis (faktorial, pangkat, deret Fibonacci, deret Collatz) dan menghasilkan pola visual (pola bintang).

4. **Aplikasi Praktis**: Menunjukkan bagaimana konsep-konsep ini dapat diterapkan dalam skenario nyata, seperti mencatat hasil pertandingan, mengelola data mahasiswa, menentukan pemenang kompetisi, mengecek palindrom, hingga perhitungan sederhana seperti gaji lembur atau status kelulusan.

```
PS C:\Users\HP\OneDrive\modul7> go run "c:\Users\HP\OneDrive\modul7\con
toh1\1.go"
Masukkan jumlah tanaman: 4
Masukkan tinggi tanaman ke-1 (cm): 10
Masukkan tinggi tanaman ke-2 (cm): 9.5
Masukkan tinggi tanaman ke-3 (cm): 5
Masukkan tinggi tanaman ke-4 (cm): 8

Tinggi tanaman tertinggi: 10.00 cm
Tinggi tanaman terpendek: 9.50 cm
```



```go
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)
}
```

Penjelasan: Kumpulan kode Go ini merupakan demonstrasi fundamental tentang dasar-dasar pemrograman Go dan penerapannya. Intinya, kode-kode ini menampilkan:

1. **Struktur dan Organisasi Kode**: Menekankan struktur standar Go (package main, func main()) serta modularitas melalui **fungsi** (mengembalikan nilai) dan **prosedur** (tidak mengembalikan nilai) untuk menciptakan kode yang rapi dan terorganisir.

2. **Manajemen Data**: Mengilustrasikan penggunaan **array** (ukuran tetap) dan **slice** (ukuran dinamis) untuk menyimpan dan memanipulasi data, termasuk operasi dasar seperti pengisian, pencarian nilai ekstrem (min/max), penghapusan elemen, dan perhitungan statistik (rata-rata, standar deviasi, frekuensi). Ini juga mencakup penggunaan tipe data kustom seperti struct dan alias untuk representasi data yang lebih kompleks.

3. **Implementasi Algoritma**: Membandingkan dan menerapkan dua pendekatan utama dalam pemecahan masalah:

    - **Rekursi**: Fungsi yang memanggil dirinya sendiri, dengan base case sebagai kondisi berhenti untuk mencegah *infinite loop*.

    - **Iterasi**: Menggunakan perulangan untuk tugas-tugas berulang. Keduanya digunakan untuk menyelesaikan masalah matematis (perpangkatan, faktorial, deret Fibonacci, deret Collatz) dan menghasilkan pola visual (pola bintang).

4. **Aplikasi Praktis**: Menunjukkan bagaimana konsep-konsep pemrograman ini dapat diaplikasikan dalam skenario dunia nyata, seperti simulasi pencatatan pertandingan olahraga, pengelolaan data mahasiswa, kompetisi skor, pengecekan palindrom, hingga perhitungan sederhana seperti gaji lembur dan status kelulusan.

```
PS C:\Users\HP\OneDrive\modul7> go run "c:\Users\HP\OneDrive\modul7\contoh2\2.go"
Masukkan jumlah buku dan jumlah buku per rak: 2 3

Masukkan harga setiap buku (dalam ribuan Rp):
5000
7000

Rata-rata harga per rak: 4000.00
Harga termahal: 7000.00 Rp
5000
7000

Rata-rata harga per rak: 4000.00
Harga termahal: 7000.00 Rp
Harga termurah: 5000.00 Rp
```

```go
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: %.2f\n", rataRata)
    fmt.Printf("Jumlah siswa di atas rata-rata: %d\n", diAtasRataRata)
}
```

Penjelasan:

```
PS C:\Users\HP\OneDrive\modul7> go run "c:\Users\HP\OneDrive\modul7\contoh3\3.go"
Masukkan jumlah siswa: 5

Masukkan nilai ujian masing-masing siswa:
90
85
100
80
95

Nilai terendah: 80
Nilai tertinggi: 100
Rata-rata kelas: 90.00
Jumlah siswa di atas rata-rata: 2
```

III. UNGUIDED

```go
package main

import "fmt"

func main() {
    fmt.Println("Program Pencatatan Berat Kelinci")
    fmt.Println("============================")

    var jumlahKelinci int
    fmt.Println("\nNAMA: MULIA AKBAR NANDA PRATAMA")
    fmt.Println("NIM: 103112400034")
    fmt.Print("\nMasukkan jumlah kelinci: ")
    fmt.Scan(&jumlahKelinci)

    if jumlahKelinci <= 0 {
        fmt.Println("Error: Jumlah kelinci harus lebih dari 0!")
        return
    }

    daftarBerat := make([]float64, jumlahKelinci)
    var totalBerat float64

    fmt.Println("\nMasukkan berat kelinci (dalam kg):")
    for i := 0; i < jumlahKelinci; i++ {
        fmt.Printf("Berat kelinci ke-%d: ", i+1)
        fmt.Scan(&daftarBerat[i])

        if daftarBerat[i] <= 0 {
            fmt.Println("Error: Berat kelinci harus lebih dari 0!")
            return
        }
        totalBerat += daftarBerat[i]
    }

    paling Ringan := daftarBerat[0]
    palingBerat := daftarBerat[0]
    indexRingan := 0
    indexBerat := 0

    for i := 1; i < jumlahKelinci; i++ {
        if daftarBerat[i] < palingRingan {
            palingRingan = daftarBerat[i]
            indexRingan = i
        }
        if daftarBerat[i] > palingBerat {
            palingBerat = daftarBerat[i]
            indexBerat = i
        }
    }

    fmt.Println("\nHasil Analisis Berat Kelinci")
    fmt.Println("==========================")
    fmt.Printf("Jumlah kelinci: %d\n", jumlahKelinci)
    fmt.Printf("Rata-rata berat: %.2f kg\n", totalBerat/float64(jumlahKelinci))
    fmt.Printf("Kelinci paling ringan: Kelinci ke-%d (%.2f kg)\n", indexRingan+1, palingRingan)
    fmt.Printf("Kelinci paling berat: Kelinci ke-%d (%.2f kg)\n", indexBerat+1, palingBerat)
}
```

Penjelasan: Kumpulan materi ini adalah pengantar komprehensif dan praktis mengenai fondasi pemrograman dengan Bahasa Go, mencakup:

1. Struktur Dasar Program**:** Setiap program Go dimulai dengan package main dan func main().
2. Interaksi I/O: Penggunaan paket fmt untuk input (Scan, Scanf) dan output (Print, Println, Printf).
3. Pengelolaan Data Koleksi:
    - Array: Untuk data dengan ukuran tetap.
    - Slice: Untuk data dengan ukuran dinamis (len, cap, append).
4. Kontrol Alur Program: Penggunaan perulangan (for) dan kondisional (if-else if-else) untuk mengontrol jalannya program.
5. Modularitas Kode:
    - Prosedur**:** Unit kode yang melakukan serangkaian instruksi tanpa mengembalikan nilai (fokus pada *side effect*).
    - Fungsi: Unit kode yang melakukan perhitungan atau proses dan selalu mengembalikan nilai eksplisit.
6. Konsep Algoritma:
    - Pencarian nilai ekstrem (min/max).
    - Rekursi: Fungsi memanggil dirinya sendiri, memerlukan *base case* (kondisi berhenti) dan *recursive case* (pemanggilan diri).
    - Iterasi: Pendekatan menggunakan perulangan sebagai alternatif rekursi.
7. Tipe Data Lanjut: Penggunaan type (alias) dan struct untuk mendefinisikan tipe data kustom yang lebih kompleks dan terstruktur.

```
PS C:\Users\HP\OneDrive\modul7> go run "c:\Users\HP\OneDrive\modul7\soal1\1.go"
Program Pencatatan Berat Kelinci
==============================

Masukkan jumlah kelinci: 4

Masukkan berat kelinci (dalam kg):
Berat kelinci ke-1: 10
Berat kelinci ke-2: 3
Berat kelinci ke-3: 5.5
Berat kelinci ke-4: 2.4

Hasil Analisis Berat Kelinci
==========================
Jumlah kelinci: 4
Rata-rata berat: 5.22 kg
Kelinci paling ringan: Kelinci ke-4 (2.40 kg)
Kelinci paling berat: Kelinci ke-1 (10.00 kg)
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Program Distribusi Ikan ke Wadah")
    fmt.Println("==============================")

    var jumlahIkan, kapasitasWadah int
    var berat [1000]float64

    fmt.Print("\nMasukkan jumlah ikan: ")
    fmt.Scan(&jumlahIkan)

    fmt.Print("Masukkan kapasitas per wadah: ")
    fmt.Scan(&kapasitasWadah)

    if jumlahIkan <= 0 || jumlahIkan > 1000 || kapasitasWadah <= 0 {
        fmt.Println("Error: Input tidak valid!")
        fmt.Println("- Jumlah ikan harus antara 1-1000")
        fmt.Println("- Kapasitas wadah harus lebih dari 0")
        return
    }

    fmt.Println("\nMasukkan berat ikan (dalam kg):")
    for i := 0; i < jumlahIkan; i++ {
        fmt.Printf("Berat ikan ke-%d: ", i+1)
        fmt.Scan(&berat[i])

        if berat[i] <= 0 {
            fmt.Println("Error: Berat ikan harus lebih dari 0!")
            return
        }
    }

    jumlahWadah := (jumlahIkan + kapasitasWadah - 1) / kapasitasWadah
    totalBerat := make([]float64, jumlahWadah)

    for i := 0; i < jumlahIkan; i++ {
        wadah := i / kapasitasWadah
        totalBerat[wadah] += berat[i]
    }

    fmt.Println("\nHasil Distribusi Ikan")
    fmt.Println("====================")
    fmt.Printf("Jumlah ikan: %d\n", jumlahIkan)
    fmt.Printf("Kapasitas per wadah: %d\n", kapasitasWadah)
    fmt.Printf("Jumlah wadah: %d\n\n", jumlahWadah)

    fmt.Println("Berat Ikan per Wadah:")
    for i := 0; i < jumlahWadah; i++ {
        fmt.Printf("Wadah %d: %.2f kg\n", i+1, totalBerat[i])
    }

    var total float64
    for i := 0; i < jumlahWadah; i++ {
        total += totalBerat[i]
    }

    fmt.Printf("\nRata-rata berat per wadah: %.2f kg\n", total/float64(jumlahWadah))
}
```

Penjelasan: Program ini membekali pemahaman dasar Go dari sintaks hingga pola algoritma, dengan penekasan pada modularitas, pengelolaan data, dan penanganan logika program.

```
PS C:\Users\HP\OneDrive\modul7> go run "c:\Users\HP\OneDrive\modul7\soal2\2.go"
Program Distribusi Ikan ke Wadah
============================

Masukkan jumlah ikan: 5
Masukkan kapasitas per wadah: 3

Masukkan berat ikan (dalam kg):
Berat ikan ke-1: 1.0
Berat ikan ke-2: 2
Berat ikan ke-3: 4.5
Berat ikan ke-4: 2.3
Berat ikan ke-5: 2

Hasil Distribusi Ikan
====================
Jumlah ikan: 5
Kapasitas per wadah: 3
Jumlah wadah: 2

Berat Ikan per Wadah:
Wadah 1: 7.50 kg
Wadah 2: 4.30 kg

Rata-rata berat per wadah: 5.90 kg
```

```go
package main

import "fmt"

func hitungMinMax(arr []float64, bMin *float64, bMax *float64) {
    *bMin = arr[0]
    *bMax = arr[0]

    for i := 1; i < len(arr); i++ {
        if arr[i] < *bMin {
            *bMin = arr[i]
        }
        if arr[i] > *bMax {
            *bMax = arr[i]
        }
    }
}

func hitungRerata(arr []float64) float64 {
    total := 0.0
    for _, v := range arr {
        total += v
    }
    return total / float64(len(arr))
}

func main() {
    fmt.Println("Program Analisis Berat Badan Balita")
    fmt.Println("--------------------------------")

    var jumlahBalita int
    fmt.Print("\nMasukkan jumlah balita: ")
    fmt.Scan(&jumlahBalita)

    if jumlahBalita <= 0 {
        fmt.Println("Error: Jumlah balita harus lebih dari 0!")
        return
    }

    beratBalita := make([]float64, jumlahBalita)

    fmt.Println("\nMasukkan berat badan balita (dalam kg):")
    for i := 0; i < jumlahBalita; i++ {
        fmt.Printf("Berat balita ke-%d: ", i+1)
        fmt.Scan(&beratBalita[i])

        if beratBalita[i] <= 0 {
            fmt.Println("Error: Berat badan harus lebih dari 0!")
            return
        }
    }

    var min, max float64
    hitungMinMax(beratBalita, &min, &max)
    rerata := hitungRerata(beratBalita)

    fmt.Println("\nHasil Analisis Berat Badan Balita")
    fmt.Println("--------------------------------")
    fmt.Printf("Jumlah balita: %d\n", jumlahBalita)
    fmt.Printf("Berat minimum: %.2f kg\n", min)
    fmt.Printf("Berat maksimum: %.2f kg\n", max)
    fmt.Printf("Rata-rata berat: %.2f kg\n", rerata)
    var diBawahRerata, diAtasRerata int
    for _, berat := range beratBalita {
        if berat < rerata {
            diBawahRerata++
        } else if berat > rerata {
            diAtasRerata++
        }
    }

    fmt.Printf("\nBalita dengan berat di bawah rata-rata: %d\n", diBawahRerata)
    fmt.Printf("Balita dengan berat di atas rata-rata: %d\n", diAtasRerata)
}
```

Penjelasan: Program ini bekerja dalam beberapa langkah utama:

1. **Input Data:** Meminta pengguna memasukkan jumlah balita, kemudian berat badan masing-masing balita. Ada validasi input untuk memastikan jumlah balita positif dan berat badan juga positif.
2. **Perhitungan Statistik Dasar:**
    - **Rata-rata:** Menjumlahkan semua berat badan, lalu membaginya dengan jumlah balita.
    - **Minimum & Maksimum:** Mengiterasi melalui semua berat badan untuk menemukan nilai terendah dan tertinggi yang dicatat.
3. **Analisis Lebih Lanjut:**
    - Menghitung berapa banyak balita yang berat badannya berada **di bawah** rata-rata.
    - Menghitung berapa banyak balita yang berat badannya berada **di atas** rata-rata.
4. **Output Hasil:** Menampilkan semua hasil perhitungan: rata-rata berat badan, berat badan minimum dan maksimum, serta jumlah balita di bawah dan di atas rata-rata.

Singkatnya, program ini mengambil daftar berat badan balita, melakukan perhitungan statistik dasar (rata-rata, min, max), dan kemudian menganalisis distribusi berat badan relatif terhadap rata-rata.

```
PS C:\Users\HP\OneDrive\modul7> go run "c:\Users\HP\OneDrive\modul7\soal3\3.go"
Program Analisis Berat Badan Balita
==================================

Masukkan jumlah balita: 4

Masukkan berat badan balita (dalam kg):
Berat balita ke-1: 5.3
Berat balita ke-2: 6.2
Berat balita ke-3: 4.1
Berat balita ke-4: 9.9

Hasil Analisis Berat Badan Balita
==================================
Jumlah balita: 4
Berat minimum: 4.10 kg
Berat maksimum: 9.90 kg
Rata-rata berat: 6.38 kg

Balita dengan berat di bawah rata-rata: 3
Balita dengan berat di atas rata-rata: 1
```

IV. KESIMPULAN

V. REFERENSI